



BLDEA'S S.B. ARTS AND K.C.P. SCIENCE

COLLEGE, VIJAYAPUR

(Affiliated to Rani Channamma University, Belagavi)

Practical's on DSC I

ALGEBRA-I AND CALCULUS-I

(AS PER NEP-2020)

MANUAL

B. Sc. I Semester

Name:.....

Class:.....

UUCMS No:.....

DEPARTMENT OF MATHEMATICS

Theory Based Practical's On Algebra-I and Calculus-I

Lab Practical's:

Part A: (SciLab)

Introduction to the software and commands related to the topic.

1. Computation of addition and subtraction of matrices.
2. Computation of Multiplication of matrices.
3. Computation of Trace and Transpose of Matrix.
4. Computation of Rank of matrix and Row reduced Echelon form.
5. Computation of Inverse of a Matrix using Cayley-Hamilton theorem.
6. Solving the system of homogeneous and non-homogeneous linear algebraic equations.

Part B: (Maxima)

7. Finding the nth Derivative of e^{ax} , trigonometric and hyperbolic functions.
8. Finding the nth Derivative of algebraic and logarithmic functions.
9. Finding the nth Derivative of $e^{ax+b}\sin(bx + c)$, $e^{ax+b}\cos(bx + c)$.
10. Finding the Taylor's and Maclaurin's expansions of the given functions.
11. Finding the angle between the radius vector and tangent.
12. Finding the curvatures of the given curves.
13. Tracing of standard curves (Cartesian, polar and parametric).

Program 1

Computation of addition and subtraction of matrices

```

clc
A=input("Enter the first matrix A= ")
[m,n]=size(A)
disp(A)
B=input("Enter the second matrix B=")
[k,l]=size(B)
disp(B)
if size(A)==size(B) then
C=A+B
D=A-B
disp("Addition of two matrices C=",C)
disp("Sub of two matrices D=",D)
else
disp("Addition and Subtraction of matrices
not defined")
end

```

Question1: Find the addition and subtraction of matrices of

$$\begin{bmatrix} 5 & 6 & 7 \\ 1 & 2 & 3 \\ 7 & 3 & 5 \end{bmatrix} \text{ and } \begin{bmatrix} 5 & 2 & 1 \\ 6 & 7 & 8 \\ 5 & 2 & 1 \end{bmatrix}.$$

Output 1:

Enter the first matrix A= [5 6 7; 1 2 3; 7 3 5]

5. 6. 7.

1. 2. 3.

7. 3. 5.

Enter the second matrix B= [5 2 1; 6 7 8; 5 2 1]

5. 2. 1.

6. 7. 8.

5. 2. 1.

"Addition of two matrices C="

10. 8. 8.

7. 9. 11.

12. 5. 6.

"Sub of two matrices D="

0. 4. 6.

-5. -5. -5.

2. 1. 4.

Question 2: Find the addition and subtraction of matrices of

$$\begin{bmatrix} 1 & 5 & 9 \\ 7 & 5 & 3 \\ 6 & 8 & 4 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & 5 & 3 \\ 7 & 6 & 9 \end{bmatrix}.$$

Output 2:

Enter the first matrix A= [1 5 9; 7 5 3; 6 8 4]

1. 5. 9.

7. 5. 3.

6. 8. 4.

Enter the second matrix B= [1 5 3; 7 6 9]

1. 5. 3.

7. 6. 9.

"Addition and Subtraction of matrices not defined"

Program 2

Computation of Multiplication of matrices

```

clc
A=input("Enter the first matrix A=")
[m,n]=size(A)
disp(A)
B=input("Enter the second matrix B=")
[k,l]=size(B)
disp(B)
if n==k then
C=A*B
disp("Multiplication of two matrices C=",C)
else
disp("Matrices of invalid order")
end

```

Question1: Find the multiplication of matrices of $\begin{bmatrix} 1 & 2 & 5 \\ 8 & 6 & 7 \\ 9 & 8 & 10 \end{bmatrix}$ and

$$\begin{bmatrix} 6 & 8 & 4 \\ 5 & 9 & 8 \\ 5 & 11 & 12 \end{bmatrix}.$$

Output 1:

Enter the first matrix A=[1 2 5; 8 6 7; 9 8 10]

1. 2. 5.

8. 6. 7.

9. 8. 10.

Enter the second matrix B=[6 8 4; 5 9 8; 5 11 12]

6. 8. 4.

5. 9. 8.

5. 11. 12.

"Multiplication of two matrices C="

41. 81. 80.

113. 195. 164.

144. 254. 220.

Question 2: Find the multiplication of matrices of $\begin{bmatrix} 1 & 5 & 6 \\ 6 & 8 & 2 \end{bmatrix}$ and $\begin{bmatrix} 5 & 9 & 6 \\ 5 & 8 & 7 \end{bmatrix}$.

Output 2:

Enter the first matrix A=[1 5 6; 6 8 2]

1. 5. 6.

6. 8. 2.

Enter the second matrix B=[5 9 6; 5 8 7]

5. 9. 6.

5. 8. 7.

"Matrices of invalid order"

Program 3

Computation of Trace and Transpose of Matrix

```

clc
A=input("Enter the matrix A=")
disp(A)
[m,n]=size(A)
if m==n then
    trace=0
    for i=1:m
        trace=trace+A(i,i)
    end
    disp("Trace of A=",trace)
else
    disp("Invalid order")
end

```

Question 1: Find the trace of matrix $\begin{bmatrix} 5 & 6 & 9 \\ 8 & 6 & 7 \\ 12 & 13 & 16 \end{bmatrix}$.

Output 1:

Enter the matrix A=[5 6 9; 8 6 7; 12 13 16]

```

5.   6.   9.
8.   6.   7.
12.  13.  16.

```

"Trace of A="

27.

Question 2: Find the trace of matrix $\begin{bmatrix} 1 & 5 & 6 \\ 12 & 18 & 17 \end{bmatrix}$.

Output 2:

Enter the matrix A=[1 5 6; 12 18 17]

```

1.   5.   6.
12.  18.  17.

```

"Invalid order"

```

clc
A=input("Enter Matrix A=")
disp(A)
[m,n]=size(A)
for i=1:n
    for j=1:m
        B(i,j)=A(j,i)
    end
end
disp("Transpose of A =",B)

```

Question1: Find the trace of matrix $\begin{bmatrix} 7 & 8 & 6 \\ 5 & 9 & 3 \\ 7 & 5 & 4 \end{bmatrix}$.

Output 1

Enter Matrix A=[7 8 6; 5 9 3; 7 5 4]

```

7.  8.  6.
5.  9.  3.
7.  5.  4.

```

"Transpose of A ="

```

7.  5.  7.
8.  9.  5.
6.  3.  4.

```

Program 4

Computation of Rank of matrix and Row reduced Echelon form

Rank of Matrix:

```
clc
```

```
A=input("Enter the matrix A=")
```

```
[m,n]=size(A)
```

```
disp(A)
```

```
B=rank(A)
```

```
disp("Rank of A is",B)
```

Question 1: Find the rank of matrix $\begin{bmatrix} 1 & 5 & 6 \\ 7 & 5 & 3 \\ 7 & 6 & 4 \end{bmatrix}$.

Output 1

Enter the matrix A=[1 5 6; 7 5 3; 7 6 4]

```
1. 5. 6.
7. 5. 3.
7. 6. 4.
```

"Rank of A is"

3.

Question 2: Find the rank of matrix $\begin{bmatrix} 1 & 5 & 6 \\ 7 & 5 & 3 \end{bmatrix}$.

Output 2

Enter the matrix A=[1 5 6; 7 5 3]

```
1. 5. 6.
7. 5. 3.
```

"Rank of A is"

2.

Row Reduced Echelon Form:

```

clc
A=input("Enter matrix A=")
[m,n]=size(A)
disp(A)
B=rref(A)
disp("Row Reduced Echelon Form of A is",B)

```

Output 1

Enter matrix A=[1 2 3; 7 8 6; 4 8 6]

```

1.  2.  3.
7.  8.  6.
4.  8.  6.

```

"Row Reduced Echelon Form of A is"

```

1.  0.  0.
0.  1.  0.
0.  0.  1.

```

Output 2

Enter matrix A=[2 3 5 4; 0 2 3 4; 4 8 13 12]

```

2.  3.  5.  4.
0.  2.  3.  4.
4.  8.  13. 12.

```

"Row Reduced Echelon Form of A is"

```

1.  0.  0.25 -1.
0.  1.  1.5  2.
0.  0.  0.   0.

```

Program 5

Computation of Inverse of a Matrix using Cayley-Hamilton theorem.

```

clc
A=input("Enter matrix A =")
[m,n]=size(A)
if m==n then
p=poly(A,'x')
disp("Characteristic equation of A=",p)
eg=spec(A)
disp("Eigen values of A=",eg)
adj=det(A)*inv(A)
disp("Adjoint of A=",adj)
B=(1/det(A))*(A^(n-1)*trace(A)*A+trace(adj)*eye(n,n))
disp("Inverse of A=",B)
else
disp("Invalid order")
end

```

Question 1: Obtain the inverse of matrix $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix}$ using Cayley

Hamilton theorem.

Output 1:

Enter matrix A =[1 1 1; 1 2 3; 1 3 4]

"Characteristic equation of A="

1 +3x -7x² +x³

"Eigen values of A="

-0.2184795

0.7024336

6.5160459

"Adjoint of A="

-1. -1. 1.
 -1. 3. -2.
 1. -2. 1.
"Inverse of A="
 -122. -273. -371.
 -273. -640. -868.
 -371. -868. -1186.

Question 2: Obtain the inverse of matrix $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}$ using Cayley Hamilton theorem.

Output 2:

Enter matrix A =[1 1 1; 1 2 3]

"Invalid order"

Program 6

Solving the system of homogeneous and non-homogeneous linear algebraic equations.

System of Homogeneous Linear Algebraic Equations:

```
clc
A=input("Enter the coefficient matrix A")
disp(A)
detA=det(A)
disp("Determinant of A=",detA)
if det(A)~=0 then
disp("A has trivial solution")
else
if det(A)==0 then
disp("A has infinite number of solution")
end
end
```

Question 1: Find the solution of system of homogeneous equation $x - 2y + z = 0$, $x - 2y - z = 0$, $2x - 4y - 5z = 0$.

Output 1:

Enter the coefficient matrix A[1 -2 1; 1 -2 -1; 2 -4 -5]

1. -2. 1.

1. -2. -1.

2. -4. -5.

"Determinant of A="

0.

"A has infinite number of solution"

Question 2: Find the solution of system of homogeneous equation $x + y + 3z = 0$, $3x + 4y + 4z = 0$, $7x + 16y + 12z = 0$.

Output 2:

Enter the coefficient matrix A[1 1 3; 3 4 4; 7 16 12]

1. 1. 3.
 3. 4. 4.
 7. 16. 12.

"Determinant of A="

36.000000

"A has trivial solution"

System of Non-Homogeneous Linear Algebraic Equations:

`clc`

`A=input("Enter the coefficient matrix A")`

`disp(A)`

`[m,n]=size(A)`

`B=input("Enter the matrix of constants B")`

`disp(B)`

`r1=rank(A)`

`disp("Rank of A=",r1)`

`r2=rank([AB])`

`disp("Rank of [A-B]=",r2)`

`if(m>=n&r2<n)then`

`disp("System is consistent and has infinitely many solutions")`

`else`

`if(m>=n&r1==r2)then`

`disp("System is consistent and has unique solution")`

`elseif(r1~=r2)then`

`disp("System is inconsistent and has no solution")`

`elseif(m<n)then`

`disp("System has infinitely many solutions")`

`end`

`end`

```

end
end
x=inv(A)*B
disp("Solution of the system",x)

```

Question 1: Find the solution of system of non-homogeneous equation $10x + 5y + 2z = 10$, $5x + 2y + 3z = 5$, $6x + 7y + 4z = 5$.

Output 1:

Enter the coefficient matrix A[10 5 2; 5 2 3; 6 7 4]

10. 5. 2.

5. 2. 3.

6. 7. 4.

Enter the matrix of constants B[10; 5; 5]

10.

5.

5.

"Rank of A="

3.

"Rank of [A-B]="

3.

"System is consistent and has unique solution"

"Solution of the system"

1.1170213

-0.212766

-0.0531915

Question 2: Find the solution of system of non-homogeneous equation $x + 2y + z = 2$, $2x + 4y + 3z = 3$, $3x + 6y + 5z = 4$.

Output 2:

Enter the coefficient matrix A[1 2 1; 2 4 3; 3 6 5]

1. 2. 1.

2. 4. 3.

3. 6. 5.

Enter the matrix of constants B[2; 3; 4]

2.

3.

4.

"Rank of A="

2.

"Rank of $[A-B]=$ "

2.

"System is consistent and has infinitely many solutions"

Program 7

Finding the n th Derivative of e^{ax} , trigonometric and hyperbolic functions.

Question 1: Write a maxima program to find the n th derivative of e^{ax} .

Program:

```
kill(all)$
y:%e^(a*x)$
n:4$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
```

Output:

```
"The" "1" "differentiation of " "%e^(a*x)" "is" " "a*%e^(a*x)" "
"The" "2" "differentiation of " "%e^(a*x)" "is" " "a^2*%e^(a*x)" "
"The" "3" "differentiation of " "%e^(a*x)" "is" " "a^3*%e^(a*x)" "
"The" "4" "differentiation of " "%e^(a*x)" "is" " "a^4*%e^(a*x)" "
```

```
(%i3) kill(all)$
y:%e^(a*x)$
n:4$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
```

```
The 1 differentiation of %ea x is a %ea x
The 2 differentiation of %ea x is a2 %ea x
The 3 differentiation of %ea x is a3 %ea x
The 4 differentiation of %ea x is a4 %ea x
```

Question 2: Write a maxima program to find the n th derivative of $\sin(ax + b)$.

Program:

```
kill(all)$
```

```

y:sin(a*x+b)$
n:5$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$

```

Output:

```

"The" "1" "differentiation of " "sin(a*x+b)" "is" " "a*cos(a*x+b)" "
"The" "2" "differentiation of " "sin(a*x+b)" "is" " "-
a^2*sin(a*x+b)" "
"The" "3" "differentiation of " "sin(a*x+b)" "is" " "-
a^3*cos(a*x+b)" "
"The" "4" "differentiation of " "sin(a*x+b)" "is" " "a^4*sin(a*x+b)"
"
"The" "5" "differentiation of " "sin(a*x+b)" "is" " "a^5*cos(a*x+b)"
"

```

```

(%i3) kill(all)$
y:sin(a*x+b)$
n:5$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
The 1 differentiation of sin(a x+b) is a cos(a x+b)
The 2 differentiation of sin(a x+b) is -a^2 sin(a x+b)
The 3 differentiation of sin(a x+b) is -a^3 cos(a x+b)
The 4 differentiation of sin(a x+b) is a^4 sin(a x+b)
The 5 differentiation of sin(a x+b) is a^5 cos(a x+b)

```

Examples: Write a maxima program to find the nth derivative of the following:

1. $\sinh(ax + b)$
2. $\cosh(ax + b)$

Program 8

Finding the nth Derivative of algebraic and logarithmic functions.

Question 1: Write a maxima program to find the nth derivative of algebraic function $(ax + b)^m$.

Program:

```
kill(all)$
y:(a*x+b)^(m)$
n:5$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
```

Output:

```
"The" "1" "differentiation of " "(a*x+b)^m" "is "
"a*m*(a*x+b)^(m-1)" "
"The" "2" "differentiation of " "(a*x+b)^m" "is " "a^2*(m-
1)*m*(a*x+b)^(m-2)" "
"The" "3" "differentiation of " "(a*x+b)^m" "is " "a^3*(m-2)*(m-
1)*m*(a*x+b)^(m-3)" "
"The" "4" "differentiation of " "(a*x+b)^m" "is " "a^4*(m-3)*(m-
2)*(m-1)*m*(a*x+b)^(m-4)" "
"The" "5" "differentiation of " "(a*x+b)^m" "is " "a^5*(m-4)*(m-
3)*(m-2)*(m-1)*m*(a*x+b)^(m-5)" "
```

```
(%i3) kill(all)$
y:(a*x+b)^(m)$
n:5$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
```

```
The 1 differentiation of (a x+b)^m is a m (a x+b)^(m-1)
The 2 differentiation of (a x+b)^m is a^2 (m-1) m (a x+b)^(m-2)
The 3 differentiation of (a x+b)^m is a^3 (m-2) (m-1) m (a x+b)^(m-3)
The 4 differentiation of (a x+b)^m is a^4 (m-3) (m-2) (m-1) m (a x+b)^(m-4)
The 5 differentiation of (a x+b)^m is a^5 (m-4) (m-3) (m-2) (m-1) m (a x+b)^(m-5)
```

Question 2: Write a maxima program to find the n th derivative of logarithmic function $\log(ax + b)$.

Program:

```
kill(all)$
y:log(a*x+b)$
n:5$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
```

Output:

```
"The" "1" "differentiation of " "log(a*x+b)" "is" "a/(a*x+b)" "
"The" "2" "differentiation of " "log(a*x+b)" "is" "-a^2/(a*x+b)^2" "
"The" "3" "differentiation of " "log(a*x+b)" "is" "
(2*a^3)/(a*x+b)^3" "
"The" "4" "differentiation of " "log(a*x+b)" "is" "-
(6*a^4)/(a*x+b)^4" "
"The" "5" "differentiation of " "log(a*x+b)" "is" "
(24*a^5)/(a*x+b)^5" "
```

```
(%i3) kill(all)$
y:log(a*x+b)$
n:5$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
```

```
The 1 differentiation of log(a x+b) is  $\frac{a}{a x+b}$ 
The 2 differentiation of log(a x+b) is  $-\frac{a^2}{(a x+b)^2}$ 
The 3 differentiation of log(a x+b) is  $\frac{2 a^3}{(a x+b)^3}$ 
The 4 differentiation of log(a x+b) is  $-\frac{6 a^4}{(a x+b)^4}$ 
The 5 differentiation of log(a x+b) is  $\frac{24 a^5}{(a x+b)^5}$ 
```


Program 9

Finding the nth Derivative of $e^{ax}\sin(bx + c)$ and $e^{ax}\cos(bx + c)$.

Question 1: Write a maxima program to find the nth derivative of $e^{ax}\sin(bx + c)$.

Program:

```
kill(all)$
y:exp(a*x)*sin(b*x+c)$
n:3$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$
```

Output:

```
"The" "1" "differentiation of " "%e^(a*x)*sin(b*x+c)" "is " "
"a*%e^(a*x)*sin(b*x+c)+b*%e^(a*x)*cos(b*x+c)" "
"The" "2" "differentiation of " "%e^(a*x)*sin(b*x+c)" "is " "-
b^2*%e^(a*x)*sin(b*x+c)+a^2*%e^(a*x)*
sin(b*x+c)+2*a*b*%e^(a*x)*cos(b*x+c)" "
"The" "3" "differentiation of " "%e^(a*x)*sin(b*x+c)" "is " "-
3*a*b^2*%e^(a*x)*sin(b*x+c)+a^3*%e^(a*x)*
sin(b*x+c)-b^3*%e^(a*x)*cos(b*x+c)+3*a^2*b*%e^(a*x)*cos(b*x+c)"
"
```

```
(%i3) kill(all)$
y:exp(a*x)*sin(b*x+c)$
n:3$
for i:1 thru n do (
y1:diff(y,x,i),
print("The",i,"differentiation of ",y,"is ",y1))$

The 1 differentiation of  $e^{ax}\sin(bx+c)$  is  $a e^{ax}\sin(bx+c) + b e^{ax}\cos(bx+c)$ 

The 2 differentiation of  $e^{ax}\sin(bx+c)$  is  $-b^2 e^{ax}\sin(bx+c) + a^2 e^{ax}\sin(bx+c) + 2ab e^{ax}\cos(bx+c)$ 

The 3 differentiation of  $e^{ax}\sin(bx+c)$  is  $-3ab^2 e^{ax}\sin(bx+c) + a^3 e^{ax}\sin(bx+c) - b^3 e^{ax}\cos(bx+c) + 3a^2 b e^{ax}\cos(bx+c)$ 
```

Example: Write a maxima program to find the nth derivative of $e^{ax}\cos(bx + c)$.

Program 10

Finding the Taylor's and Maclaurin's expansions of the given functions.

Question 1: Write a maxima program to find the Taylor's and Maclaurin's expansions of e^x .

Program:

```
kill(all)$
y:exp(x)$
ts:taylor(y,x,2,4)$
ms:taylor(y,x,0,4)$
print("The Taylor's expansion of",y,"about x=2 is ")$
print(ts)$
print("The Maclaurian's expansion of",y,"is ")$
print(ms)$
```

Output:

```
"The Taylor's expansion of" " %e^x" "about x=2 is" " "
%e^2+%e^2*(x-2)+(%e^2*(x-2)^2)/2+(%e^2*(x-2)^3)/6+(%e^2*(x-
2)^4)/24+..." "
"The Maclaurian's expansion of" " %e^x" "is" " "
1+x+x^2/2+x^3/6+x^4/24+..." "
```

```
→ kill(all)$
y:exp(x)$
ts:taylor(y,x,2,4)$
ms:taylor(y,x,0,4)$
print("The Taylor's expansion of",y,"about x=2 is ")$
print(ts)$
print("The Maclaurian's expansion of",y,"is ")$
print(ms)$
```

The Taylor's expansion of e^x about $x=2$ is

$$e^2 + e^2(x-2) + \frac{e^2(x-2)^2}{2} + \frac{e^2(x-2)^3}{6} + \frac{e^2(x-2)^4}{24} + \dots$$

The Maclaurian's expansion of e^x is

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \dots$$

Question 2: Write a maxima program to find the Taylor's and Maclaurin's expansions of $\tan^{-1} x$.

Program:

```
kill(all)$
y:exp(x)$
ts:taylor(y,x,3,6)$
ms:taylor(y,x,0,6)$
print("The Taylor's expansion of",y,"about x=3 is ")$
print(ts)$
print("The Maclaurian's expansion of",y,"is ")$
print(ms)$
```

Output:

```
"The Taylor's expansion of" "atan(x)" "about x=3 is " "
atan(3)+(x-3)/10-(3*(x-3)^2)/100+(13*(x-3)^3)/1500-(3*(x-
3)^4)/1250+(79*(x-3)^5)/125000-(39*(x-3)^6)/250000+..." "
"The Maclaurian's expansion of" "atan(x)" "is " "
x-x^3/3+x^5/5+..." "
```

```
(%i7) kill(all)$
y:atan(x)$
ts:taylor(y,x,3,6)$
ms:taylor(y,x,0,6)$
print("The Taylor's expansion of",y,"about x=3 is ")$
print(ts)$
print("The Maclaurian's expansion of",y,"is ")$
print(ms)$

The Taylor's expansion of atan(x) about x=3 is
atan(3) +  $\frac{x-3}{10} - \frac{3(x-3)^2}{100} + \frac{13(x-3)^3}{1500} - \frac{3(x-3)^4}{1250} + \frac{79(x-3)^5}{125000} - \frac{39(x-3)^6}{250000} + \dots$ 

The Maclaurian's expansion of atan(x) is
 $x - \frac{x^3}{3} + \frac{x^5}{5} + \dots$ 
```

Example: Write a maxima program to find the Taylor's and Maclaurin's expansions of 1. $\log(x)$ 2. $\sin^{-1}x$ 3. $\log(\sec x)$ 4. $\frac{1}{1+x}$

Program 11

Finding the angle between the radius vector and tangent.

Question 1: Write a maxima program to find the angle between the radius vector and tangent $r = a(1 - \cos t)$.

Program:

```
kill(all)$
a:1$
r: a*(1- cos(t))$
d:diff(r,t)$
b:r*(1/d)$
ta:trigreduce(trigrat(b))$
print("The tan(angle) =",ta)$
print("The angle between the radius vector and tangent is
",atan(ta))$
```

Output:

```
"The tan(angle) =" "tan(t/2)" "
"The angle between the radius vector and
tangent is " "atan(tan(t/2))" "
```

```
(%i7) kill(all)$
a:1$
r: a*(1- cos(t))$
d:diff(r,t)$
b:r*(1/d)$
ta:trigreduce(trigrat(b))$
print("The tan(angle) =",ta)$
print("The angle between the radius vector and tangent is ",atan(ta))$
```

The $\tan(\text{angle}) = \tan\left(\frac{t}{2}\right)$

The angle between the radius vector and tangent is $\text{atan}\left(\tan\left(\frac{t}{2}\right)\right)$

Question 2: Write a maxima program to find the angle between the radius vector and tangent $r = at$.

Question 3: Write a maxima program to find the angle between the radius vector and tangent $r = a(1 + \cos t)$.

Program:

```
kill(all)$
a:1$
r(t):= a*(1+cos(t))$
dt:ratsimp(diff(r(t),t))$
b(t):=r(t)*(1/dt)$
ta(t):=trigreduce(trigrat(b(t)))$
print("The tan(angle) =",ta(t))$
print("The angle between the radius vector and tangent is
",atan(ta(t)))$
a1:at(ta(t),[t=%pi/2])$
print("The tan(angle) =",a1)$
d:atan2(1,a1)$
print("The angle between the radius vector and tangent
at t=pi/2 is ",d)$
```

Output:

```
"The tan(angle) =" "-cot(t/2)" "
"The angle between the radius vector and tangent is "" "-
atan(cot(t/2))" "
"The tan(angle) =" "-1" "
"The angle between the radius vector and tangent at
t=pi/2 is "" "(3*%pi)/4" "
```

```
(%i1) kill(all)$
a:1$
r(t):= a*(1+cos(t))$
dt:ratsimp(diff(r(t),t))$
b(t):=r(t)*(1/dt)$
ta(t):=trigreduce(trigrat(b(t)))$
print("The tan(angle) =",ta(t))$
print("The angle between the radius vector and tangent is ",atan(ta(t)))$
a1:at(ta(t),[t=%pi/2])$
print("The tan(angle) =",a1)$
d:atan2(1,a1)$
print("The angle between the radius vector and tangent at t=pi/2 is ",d)$

The tan(angle) =  $-\cot\left(\frac{t}{2}\right)$ 

The angle between the radius vector and tangent is  $-\operatorname{atan}\left(\cot\left(\frac{t}{2}\right)\right)$ 

The tan(angle) = -1

The angle between the radius vector and tangent at t=pi/2 is  $\frac{3\pi}{4}$ 
```

Program 12

Finding the curvatures of the given curves.

Question 1: Write a maxima program to find the radius of curvature and curvature of the curve $y^2 = 4ax$ at $x = 1$ and $a = 1$.

Program:

```
kill(all)$
a:1$
y:sqrt(4*a*x)$
y1:ratsimp(diff(y,x,1))$
y2:ratsimp(diff(y,x,2))$
rc:ratsimp((1+y1^2)^(3/2)/y2)$
k:ratsimp(1/rc)$
print("Radius of curvature at any point is ",rc)$
print("Radius of curvature at x=1 is ",at(rc,[x=1]))$
print("Curvature of the curve at any point is ",k)$
print("Curvature of the curve at x=1 is ",at(k,[x=1]))$
```

Output:

```
"Radius of curvature at any point is "" "-
2*x^(3/2)*((x+1)/x)^(3/2)" "
"Radius of curvature at x=1 is "" "-2^(5/2)" "
"Curvature of the curve at any point is "" "-
1/(2*x^(3/2)*((x+1)/x)^(3/2))" "
"Curvature of the curve at x=1 is "" "-1/2^(5/2)" "
```

```
(%i10) kill(all)$
a:1$
y:sqrt(4*a*x)$
y1:ratsimp(diff(y,x,1))$
y2:ratsimp(diff(y,x,2))$
rc:ratsimp((1+y1^2)^(3/2)/y2)$
k:ratsimp(1/rc)$
print("Radius of curvature at any point is ",rc)$
print("Radius of curvature at x=1 is ",at(rc,[x=1]))$
print("Curvature of the curve at any point is ",k)$
print("Curvature of the curve at x=1 is ",at(k,[x=1]))$

Radius of curvature at any point is  $-2x^{3/2}\left(\frac{x+1}{x}\right)^{3/2}$ 
Radius of curvature at x=1 is  $-2^{5/2}$ 
Curvature of the curve at any point is  $-\frac{1}{2x^{3/2}\left(\frac{x+1}{x}\right)^{3/2}}$ 
Curvature of the curve at x=1 is  $-\frac{1}{2^{5/2}}$ 
```

Question 2: Write a maxima program to find the radius of curvature and curvature of the curve $\sqrt{x} + \sqrt{y} = 1$ at $\left(\frac{1}{4}, \frac{1}{4}\right)$.

Program:

```
kill(all)$
y:(1-sqrt(x))^2$
y1:ratsimp(diff(y,x,1))$
y2:ratsimp(diff(y,x,2))$
rc:ratsimp((1+y1^2)^(3/2)/y2)$
k:ratsimp(1/rc)$
print("Radius of curvature at any point is ",rc)$
print("Curvature of the curve at any point is ",k)$
print("Radius of curvature at x=1/4 & y=1/4 is
",at(rc,[x=1/4,y=1/4]))$
print("Curvature of the curve at (1/4,1/4) is
",at(k,[x=1/4,y=1/4]))$
```

Output:

```
"Radius of curvature at any point is "" 2*(-(-
2*x+2*sqrt(x)-1)/x)^(3/2)*x^(3/2)" "
"Curvature of the curve at any point is "" 1/(2*(-(-
2*x+2*sqrt(x)-1)/x)^(3/2)*x^(3/2))" "

"Radius of curvature at x=1/4 & y=1/4 is "" 1/sqrt(2)" "
"Curvature of the curve at (1/4,1/4) is "" sqrt(2)" "
```

```
(%i9) kill(all)$
y:(1-sqrt(x))^2$
y1:ratsimp(diff(y,x,1))$
y2:ratsimp(diff(y,x,2))$
rc:ratsimp((1+y1^2)^(3/2)/y2)$
k:ratsimp(1/rc)$
print("Radius of curvature at any point is ",rc)$
print("Curvature of the curve at any point is ",k)$
print("Radius of curvature at x=1/4 & y=1/4 is ",at(rc,[x=1/4,y=1/4]))$
print("Curvature of the curve at (1/4, 1/4) is ",at(k,[x=1/4,y=1/4]))$
```

Radius of curvature at any point is $2 \left(-\frac{-2x+2\sqrt{x}-1}{x} \right)^{3/2} x^{3/2}$

Curvature of the curve at any point is $\frac{1}{2 \left(-\frac{-2x+2\sqrt{x}-1}{x} \right)^{3/2} x^{3/2}}$

Radius of curvature at x=1/4 & y=1/4 is $\frac{1}{\sqrt{2}}$

Curvature of the curve at (1/4, 1/4) is $\sqrt{2}$

Question 3: Write a maxima program to find the radius of curvature and curvature of the curve $x^3 + y^3 - 3axy = 0$ at $\left(\frac{3a}{2}, \frac{3a}{2}\right)$.

Program:

```
kill(all)$
f(x,y):=x^3+y^3-3*a*x*y$
fx:ratsimp(diff(f(x,y),x,1))$
fxx:ratsimp(diff(f(x,y),x,2))$
fy:ratsimp(diff(f(x,y),y,1))$
fyy:ratsimp(diff(f(x,y),y,2))$
fxy:ratsimp(diff(fy,x,1))$
rc:ratsimp(-
(fx^2+fy^2)^(3/2)/(fx^2*fyy-2*fx*fxy+fy^2*fxx))$
k:ratsimp(1/rc)$
print("fx=",fx,"fy=",fy,"fxy=",fxy,"fxx=",fxx,"fyy=",fyy)$
print("Radius of curvature at x=3a/2 & y=3a/2 is ",at(rc,[x=3*a/2,y=3*a/2]))$
```



```
print("Curvature of the curve at (3a/2,3a/2) is",at(k,[x=3*a/2,y=3*a/2]))$
```

Output:

```
"fx=" "3*x^2-3*a*y" "fy=" "3*y^2-3*a*x" "fxy=" "-3*a"
"fxx=" "6*x" "fyy=" "6*y" "
"Radius of curvature at x=3a/2 &y=3a/2 is " "-
(3*a)/2^(7/2)" "
"Curvature of the curve at (3a/2,3a/2) is " "-
2^(7/2)/(3*a)" "
```

```
(%i11) kill(all)$
f(x,y):=x^3+y^3-3*a*x*y$
fx:ratsimp(diff(f(x,y),x,1))$
fxx:ratsimp(diff(f(x,y),x,2))$
fy:ratsimp(diff(f(x,y),y,1))$
fyy:ratsimp(diff(f(x,y),y,2))$
fxy:ratsimp(diff(fy,x,1))$
rc:ratsimp(-(fx^2+fy^2)^(3/2)/(fx^2*fyy-2*fx*fxy+fy^2*fxx))$
k:ratsimp(1/rc)$
print("fx=",fx,"fy=",fy,"fxy=",fxy,"fxx=",fxx,"fyy=",fyy)$
print("Radius of curvature at x=3a/2 & y=3a/2 is ",at(rc,[x=3*a/2,y=3*a/2]))$
print("Curvature of the curve at (3a/2,3a/2) is ",at(k,[x=3*a/2,y=3*a/2]))$

fx= 3 x^2 - 3 a y fy= 3 y^2 - 3 a x fxy= -3 a fxx= 6 x fyy= 6 y
Radius of curvature at x=3a/2 & y=3a/2 is - 3 a
2^(7/2)
Curvature of the curve at (3a/2,3a/2) is - 2^(7/2)
3 a
```



**BLDEA'S S.B. ARTS AND K.C.P. SCIENCE
COLLEGE, VIJAYAPUR**

(Affiliated to Rani Channamma University, Belagavi)

**Practical's on DSC 2
ALGEBRA-II AND CALCULUS-II
(AS PER NEP-2020)
MANUAL**

B. Sc. II Semester

Name:.....

Class:.....

UUCMS No:.....

DEPARTMENT OF MATHEMATICS

Theory Based Practical's On Algebra-II and Calculus-II**Lab Practical's:****Part A:**

1. Program for verification of binary operations.
2. Computation of identity and inverse elements of a group.
3. Program to construct Cayley's table and test abelian for given finite set.
4. Program to find all possible cosets of the given finite group.
5. Program to find generators and corresponding possible subgroups of a cyclic group.
6. Programs to verification of Lagrange's theorem with suitable examples.

Part B:

7. Program to verify the Euler's ϕ function for a given finite group.
8. Program to verify the Euler's theorem and its extension.
9. Programs to construct series using Maclaurin's expansion for functions of two variables.
10. Program to evaluate the line integrals with constant and variable limits.
11. Program to evaluate the Double integrals with constant and variable limits.
12. Program to evaluate the Triple integrals with constant and variable limits.

Program 1

Program for verification of Binary operation.

A binary operation $*$ in a set G is a mapping $*$: $G \times G \rightarrow G$.

Question 1: Write a maxima program to verify addition modulo 6 i.e. $G = \{0, 1, 2, 3, 4, 5\}$ is a binary operation or not.

Program:

```
kill(all)$
S:{0,1,2,3,4,5}$
bo(a,b):=mod(a+b,6)$
flag:1$
for a in S do(
for b in S do (
if not elementp(bo(a,b),S) then flag:0 ))$
if flag = 1 then
disp("Given operation is a binary operation")
else
disp("Given operation is not binary operation")$
```

Output:

"Given operation is a binary operation"

```
(%i5) kill(all)$
S:{0,1,2,3,4,5}$
bo(a,b):=mod(a+b,6)$
flag:1$
for a in S do(
for b in S do (
if not elementp(bo(a,b),S) then flag:0 ))$
if flag = 1 then
disp("Given operation is a binary operation")
else
disp("Given operation is not binary operation")$
Given operation is a binary operation
```

Examples: Verify binary operations for the following group

1. $G = \{0, 1, 2\}$ under addition modulo 3
2. $G = \{0, 1, 2, 3\}$ under addition modulo 4
3. $G = \{0, 1, 2, 3, 4, 5, 6\}$ under multiplication modulo 7

Program 2

Computation of identity and inverse elements of a group.

- An element e is called an identity element of the non-empty set G if $\forall a \in G \Rightarrow a * e = e * a = a$.
- An element a^{-1} is the inverse of an element a in G if $\forall a \in G \Rightarrow a * a^{-1} = a^{-1} * a = e$.

Question 1: Write a maxima program to find the identity and inverse element of the group $z = \{1,5,7,11\}$ under multiplication modulo 12.

Program:

```
kill(all)$
z:{1,5,7,11}$
bo(a,b):=mod(a*b,12)$
e:a$
fori in z do
(
flag:1,
for j in z do
ifbo(i,j)#j and bo(j,i)# j then
flag:0,
if flag=1 then
e:i
)
)
if e # x then
print("The identity element is",e)
else
print("The identity element does not exist")$
w: {}$
fori in z do
(
for j in z do
ifbo(i, j)=e or bo(j, i)=e then
(
print("Inverse of",i,"is",j),
w :adjoin(i,w)
```

```

)
)$
if setequalp(z,w) then
  print("Inverse law is satisfied")
else
  print("inverse law is not satisfied")$

```

Output:

```

"The identity element is" "1" "
"Inverse of" "1" " "is" "1" "
"Inverse of" "5" " "is" "5" "
"Inverse of" "7" " "is" "7" "
"Inverse of" "11" " "is" "11" "
"Inverse law is satisfied" "

```

```

(%i8) kill(all)$
z:{1,5,7,11}$
bo(a,b):=mod(a·b,12)$
e:a$
for i in z do
(
flag:1,
for j in z do
if bo(i,j)#j and bo(j,i)#j then
flag:0,
if flag=1 then
e:i
)$
if e # x then
print("The identity element is",e)
else
print("The identity element does not exist")$
w: {}$
for i in z do
(
for j in z do
if bo(i, j)=e or bo(j, i)=e then
(
print("Inverse of",i,"is",j),
w :adjoin(i,w)
)
)$
if setequalp(z,w) then
print("Inverse law is satisfied")
else
print("inverse law is not satisfied")$
The identity element is 1
Inverse of 1 is 1
Inverse of 5 is 5
Inverse of 7 is 7
Inverse of 11 is 11
Inverse law is satisfied

```

Examples: Write a maxima program to find the identity and inverse element of the following group:

1. $G = \{0,1,2,3,4\}$ under addition modulo 5.
2. $G = \{2,4,6,8\}$ under multiplication modulo 10.
3. $Z_5 - \{0\}$ under multiplication modulo 5.
4. $G = \{1,3,7,9\}$ under multiplication modulo 10.
5. $(Z_7, +_7)$.

Program 3

Program to construct Cayley's table and test abelian for given finite set.

- If G is a finite group with the binary operation $*$, the Cayley table of G is a table with rows and columns labelled by the elements of the group. If entry in the row is g and column is h , then corresponding entry of table is $g * h$.
- A group G with binary operation $*$, is said to be abelian group if $\forall a, b \in G, a * b = b * a$.
- Group is abelian if Cayley table is symmetric along its diagonal axis.

Question 1: Write a maxima program to construct Cayley's table and test abelian for $(Z_3, +_3)$

Program:

```
kill(all)$
z:{0,1,2}$
print("Given set is",z)$
n:3$
CT:zn_add_table(n)$
print("Cayley's table under addition modulo 3 is",CT)$
TCT:transpose(CT)$
print("The transpose of matrix CT is",TCT)$
if CT=TCT then
disp("Cayley table is symmetric, Group is abelian")
else
disp("Cayley table not is symmetric, Group is not
abelian")$
```

Output:

```
"Given set is" "{0,1,2}"
"Cayley's table under addition modulo 3 is" "matrix(
      [0, 1, 2],
      [1, 2, 0],
      [2, 0, 1]
)" "
```

"The transpose of matrix CT is"" "matrix(

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}$$

)" "
 "Cayley table is symmetric, Group is abelian"

```
(%i8) kill(all)$
z:{0,1,2}$
print("Given set is",z)$
n:3$
CT:zn_add_table(n)$
print("Cayley's table under addition modulo 3 is",CT)$
TCT:transpose(CT)$
print("The transpose of matrix CT is",TCT)$
if CT=TCT then
disp("Cayley table is symmetric, Group is abelian")
else
disp("Cayley table not is symmetric, Group is not abelian")$
```

Given set is {0,1,2}

Cayley's table under addition modulo 3 is
$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{pmatrix}$$

The transpose of matrix CT is
$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{pmatrix}$$

Cayley table is symmetric, Group is abelian

Examples:

1. Construct($Z_8, +_8$)
2. Construct ($Z_{12}, +_{12}$)

Program 4

Program to find all possible cosets of the given finite group.

- Let H be any subgroup of a group G and a be any element of G . Then the set, $Ha = \{ha : h \in H\}$ is called right coset of H in G generated by a and the set $aH = \{ah : h \in H\}$ is called left coset of H in G generated by a with respect to multiplicative binary operation.
- $H + a = \{h + a : h \in H\}$ is right coset and $a + H = \{a + h : h \in H\}$ is left coset of H with respect to additive binary operation.

Question 1: Write a maxima program to find all distinct left coset of the subgroup $H = \{0, 4, 8\}$ in the group $(Z_{12}, +_{12})$ and find the index.

Program:

```
kill(all)$
G:{0,1,2,3,4,5,6,7,8,9,10,11}$
H:{0,4,8}$
bo(x,y):=mod(x+y,12)$
c:{}$
for i in G do(
  s:{},
  for j in H do
    s: adjoin(bo(j,i),s),
    c: adjoin(s,c))$
print("Left cosets= ",c)$
print("The index of H in G is", cardinality(c))$
```

Output:

```
"Left cosets= "" "{0,4,8},{1,5,9},{2,6,10},{3,7,11}}"" "
"The index of H in G is"" "4" "
```

```

(%i7) kill(all)$
G:{0,1,2,3,4,5,6,7,8,9,10,11}$
H:{0,4,8}$
bo(x,y):=mod(x+y,12)$
c:{}$
for i in G do
(
s:{},
for j in H do
s: adjoin(bo(j,i),s),
c: adjoin(s,c)
)$
print("Left cosets= ",c)$
print("The index of H in G is", cardinality(c))$
Left cosets= {{0,4,8},{1,5,9},{2,6,10},{3,7,11}}
The index of H in G is 4

```

Examples:

1. Find all the rightcosets of the subgroup $H = \{0,3\}$ in the group $(Z_6, +_6)$.
2. Find all the distinct cosets of $H = \{0,3,6\}$ in $(Z_9, +_9)$.

Program 6

Programs to verification of Lagrange's theorem with suitable examples.

Lagrange's Theorem: If H is a subgroup of finite group G , then $O(H)$ divides $O(G)$.

Question 1: Write a maxima program to verification of Lagrange's theorem with $G = \{1, -1, i, -i\}$ subset $H = \{1, -1\}$ of G under multiplication.

Program:

```

kill(all)$
G:set(1, -1, %i, -%i)$
print("Group G =",G)$
H:set(1, -1)$
print("Sub group H =",H)$
n:length(G)$
print("Order of group G = O(G)=",n)$
m:length(H)$
print("Order of sub group H = O(H)=",m)$
r: mod(n,m)$
if r=0 then
  disp("O(H) divides O(G), Lagrange's theorem
verified")
else
  disp("Lagrange's theorem is not verified")$
"Group G =" " {-1,1,-%i,%i} "
"Sub group H =" " {-1,1} "
"Order of group G = O(G)=" "4" "
"Order of sub group H = O(H)=" "2" "
"O(H) divides O(G), Lagrange's theorem verified"

```

```
(%i10) kill(all)$
G:set(1, -1, %i, -%i)$
print("Group G =",G)$
H:set(1, -1)$
print("Sub group H =",H)$
n:length(G)$
print("Order of group G = O(G)=",n)$
m:length(H)$
print("Order of sub group H = O(H)=",m)$
r: mod(n,m)$
if r=0 then
disp("O(H) divides O(G), Lagrange's theorem verified")
else
disp("Lagrange's theorem is not verified")$
Group G = {-1, 1, -%i, %i}
Sub group H = {-1, 1}
Order of group G = O(G)= 4
Order of sub group H = O(H)= 2
O(H) divides O(G), Lagrange's theorem verified
```

Examples: Verify Lagrange's theorem for the following:

1. $G = \{1, -1, i, -i\}$ is a group and subset $H = \{1, i\}$ under multiplication.
2. $G = \{1, -1, i, -i\}$ is a group and subset $H = \{1, -i\}$ under multiplication.

Program 7

Program to verify the Euler's ϕ function for a given finite group.

Euler's number: Let $\phi(1) = 1$, and for any integer $n > 1$, let $\phi(n)$ denote the number of positive integers less than n and relatively prime to n .

n	1	2	3	4	5	6	7	9	10
Positive integers less than n and relatively prime to n.	1	1	1,2	1,3	1,3,4	1,5	1,2,3,4,5,6	1,2,4,5,7,8	1,3,7,9
$\phi(n)$	1	1	2	2	3	2	6	6	4

Theorem: The set of all positive integers less than n and relatively prime to n form a group under multiplication modulo n .

Example: $G = \{1, 2, 3, 4, 5, 6\}$ is a group under multiplication modulo 7.

Euler's Theorem: Let a and n be integers such that $n > 0$ and $\gcd(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Question 1: Write a maxima program to verify the Euler's ϕ function for a given finite group $G = \{1, 2, 3, 4, 5, 6\}$ under multiplication modulo 7.

Program:

```
kill(all)$
```

```
G:[1, 2, 3, 4, 5, 6]$
```

```
n:7$
```

```

flag:1$
print("Given group G=",G)$
print("n=",n)$
φ:length(G)$
print("O(G)= φ(10)=",φ)$
fori thru length(G) do(
if not mod(G[i]^φ,n)=1 then flag:0)$
if flag=1 then
print ("Euler's theorem is verified")
else
print("Euler's theorem is not verified")$

```

Output:

```

"Given group G=" "[1,2,3,4,5,6]" "
"n=" "7" "
"O(G)= φ(10)=" "6" "
"Euler's theorem is verified" "

```



```
(%i9) kill(all)$
G:[1, 2, 3, 4, 5, 6]$
n:7$
flag:1$
print("Given group G=",G)$
print("n=",n)$
phi:length(G)$
print("O(G)= phi(10)=",phi)$
for i thru length(G) do(
if not mod(G[i]^phi,n)=1 then flag:0)$
if flag=1 then
print ("Euler's theorem is verified")
else
print("Euler's theorem is not verified")$
Given group G= [1, 2, 3, 4, 5, 6]
n= 7
O(G)= phi(10)= 6
Euler's theorem is verified
```

Examples: Write a maxima program to verify the Euler's ϕ function for a given finite group

1. $G = \{1, 2, 4, 5, 7, 8\}$ under multiplication modulo 9.
2. $G = \{1, 5\}$ under multiplication modulo 6.
3. $G = \{1, 3\}$ under multiplication modulo 4.

Program 8

Program to verify the Euler's theorem and its extension.

Homogeneous Function: A function $u = f(x, y)$ is said to be a homogeneous function of degree n in x & y if it satisfies the following condition:

- (i) $f(x, y) = x^n F\left(\frac{y}{x}\right)$ for some function F .
- (ii) $f(x, y) = y^n F\left(\frac{x}{y}\right)$ for some function F .
- (iii) $f(tx, ty) = t^n f(x, y)$.

Euler's Theorem: If $u = f(x, y)$ is a homogeneous function of degree n in x & y , then

$$x \frac{\partial u}{\partial x} + y \frac{\partial u}{\partial y} = n \cdot u. \text{ That is } xu_x + yu_y = n \cdot u.$$

Question 1: Verification of Euler's theorem for given homogeneous function $u = ax^2 + 2hxy + by^2$.

Program:

```
kill(all)$
u:a*x*x+2*h*x*y+b*y*y$
print("u = ",u)$
n:2$
ux:diff(u,x,1)$
uy:diff(u,y,1)$
print("ux =",ux)$
print("uy =",uy)$
lhs:ratsimp(x*ux+y*uy)$
rhs:ratsimp(n*u)$
print("LHS =",lhs)$
print("RHS =",rhs)$
if lhs=rhs then
  print("Euler's Theorem is verified")
else
```

```
print("Euler's Theorem is not verified")$
```

Output:

```
"u = "" "b*y^2+2*h*x*y+a*x^2" "
"ux ="" "2*h*y+2*a*x" "
"uy ="" "2*b*y+2*h*x" "
"LHS ="" "2*b*y^2+4*h*x*y+2*a*x^2" "
"RHS ="" "2*b*y^2+4*h*x*y+2*a*x^2" "
(%i12) kill(all)$
u:a*x*x+2*h*x*y+b*y*y$
print("u =",u)$
n:2$
ux:diff(u,x,1)$
uy:diff(u,y,1)$
print("ux =",ux)$
print("uy =",uy)$
lhs:ratsimp(x*ux+y*uy)$
rhs:ratsimp(n*u)$
print("LHS =",lhs)$
print("RHS =",rhs)$
if lhs=rhs then
print("Euler's Theorem is verified")
else
print("Euler's Theorem is not verified")$

u =  b y2 + 2 h x y + a x2
ux =  2 h y + 2 a x
uy =  2 b y + 2 h x
LHS =  2 b y2 + 4 h x y + 2 a x2
RHS =  2 b y2 + 4 h x y + 2 a x2
Euler's Theorem is verified
```

Examples: Verification of Euler's theorem for following homogeneous functions:

1. $u = \frac{xy}{x+y}$
2. $u = \frac{x^3+y^3}{x+y}$

$$3. u = \sin^{-1} \left(\frac{x}{y} \right) + \tan^{-1} \left(\frac{y}{x} \right).$$

$$4. u = \frac{x^3 + x^2y - xy^2 + 2y^3}{x+y}$$

Euler's Extension Theorem: If $u = f(x, y)$ is a homogeneous function of degree n in x & y , then

$$x^2 \frac{\partial^2 u}{\partial x^2} + 2xy \frac{\partial^2 u}{\partial x \partial y} + y^2 \frac{\partial^2 u}{\partial y^2} = n(n-1) \cdot u.$$

$$\text{That is } x^2 u_{xx} + 2xy u_{xy} + y^2 u_{yy} = n(n-1) \cdot u.$$

Question 2: Verification of Euler's Extension theorem for given homogeneous function $u = ax^2 + 2hxy + by^2$.

Program:

```
kill(all)$
u:a*x*x+2*h*x*y+b*y*y$
print("u = ",u)$
n:2$
ux:diff(u,x,1)$
uy:diff(u,y,1)$
uxx:diff(ux,x,1)$
uyy:diff(uy,y,1)$
uxy:diff(uy,x,1)$
print("ux =",ux)$
print("uy =",uy)$
print("uxx =",uxx)$
print("uyy =",uyy)$
print("uxy =",uxy)$
lhs:ratsimp(x*x*uxx+2*x*y*uxy+y*y*uyy)$
rhs:ratsimp(n*(n-1)*u)$
print("LHS =",lhs)$
```

```

print("RHS =",rhs)$
if lhs=rhs then
print("Euler's Extension Theorem is verified")
else
print("Euler's Extension Theorem is not verified")$

```

Output:

```

"u = "" "b*y^2+2*h*x*y+a*x^2" "
"ux ="" "2*h*y+2*a*x" "
"uy ="" "2*b*y+2*h*x" "
"uxx ="" "2*a" "
"uyy ="" "2*b" "
"uxy ="" "2*h" "
"LHS ="" "2*b*y^2+4*h*x*y+2*a*x^2" "
"RHS ="" "2*b*y^2+4*h*x*y+2*a*x^2" "
"Euler's Extension Theorem is verified"" "

```

```
(%i18) kill(all)$
u:a*x^2+h*x*y+b*y^2$
print("u =",u)$
n:2$
ux:diff(u,x,1)$
uy:diff(u,y,1)$
uxx:diff(ux,x,1)$
uyy:diff(uy,y,1)$
uxy:diff(uy,x,1)$
print("ux =",ux)$
print("uy =",uy)$
print("uxx =",uxx)$
print("uyy =",uyy)$
print("uxy =",uxy)$
lhs:ratsimp(x*x*uxx+2*x*y*uxy+y*y*uyy)$
rhs:ratsimp(n*(n-1)*u)$
print("LHS =",lhs)$
print("RHS =",rhs)$
if lhs=rhs then
print("Euler's Extension Theorem is verified")
else
print("Euler's Extension Theorem is not verified")$

u = b y^2 + 2 h x y + a x^2
ux = 2 h y + 2 a x
uy = 2 b y + 2 h x
uxx = 2 a
uyy = 2 b
uxy = 2 h
LHS = 2 b y^2 + 4 h x y + 2 a x^2
RHS = 2 b y^2 + 4 h x y + 2 a x^2
Euler's Extension Theorem is verified
```

Examples: Verification of Euler's Extension theorem for following homogeneous functions:

1. $u = \frac{x^3+y^3}{x+y}$
2. $u = \sin^{-1}\left(\frac{x}{y}\right) + \tan^{-1}\left(\frac{y}{x}\right)$
3. $u = \frac{x^3+x^2y-xy^2+2y^3}{x+y}$
4. $u = x^3 + y^3$

Program 9

Program to construct series using Maclaurin's expansion for function of two variables.

Question 1: Write a maxima program to find Maclaurin's expansion of $u = e^x \cos y$.

Program:

```
kill(all)$
u:%e^(x)*(cos(y))$
print("The Maclurian's expansion of",u," is")$
s:taylor(u,[x,y],0,4)$
print(s)$
```

Output:

```
"The Maclurian's expansion of" "%e^x*cos(y)" " is" "
1+x+(x^2-y^2)/2+(x^3-3*y^2*x)/6+(x^4-
```

```
6*y^2*x^2+y^4)/24+..." "
(%i4) kill(all)$
u:%e^(x)*(cos(y))$
print("The Maclurian's expansion of",u," is")$
s:taylor(u,[x,y],0,4)$
print(s)$
```

The Maclurian's expansion of $e^x \cos(y)$ is

$$1+x+\frac{x^2-y^2}{2}+\frac{x^3-3y^2x}{6}+\frac{x^4-6y^2x^2+y^4}{24}+\dots$$

Examples: Write a maxima program to find Maclaurin's expansion for following functions:

1. $u = e^x \log(1 + y)$
2. $u = e^{(x^2+y^2)}$
3. $u = \sin(x + y)$

Program 10

Program to evaluate the line integrals with constant and variable limits.

Question 1: Write a maxima program to evaluate line integral with constant limits $\int_c y \, dx - x \, dy$ along the curve $c: y = x^2$ from $(0, 0)$ to $(1, 1)$.

Program:

```
kill(all)$
depends (y,x)$
y:x^2$
Integrand:y*diff(x,x)-x*diff(y,x)$
I:integrate (Integrand,x,0,1)$
print("Integrand =",Integrand)$
print("Integral value = ",I)$
```

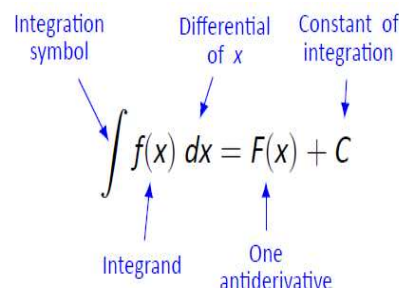
Output:

```
"Integrad ="" "-x^2" "
"Integral value"" "-1/3" "
```

```
(%i6) kill(all)$
depends (y,x)$
y:x^2$
Integrand:y·diff(x,x)−x·diff(y,x)$
I:integrate (Integrand,x,0,1)$
print("Integrand =",Integrand)$
print("Integral value = ",I)$
```

$$\text{Integrad} = -x^2$$

$$\text{Integral value} = -\frac{1}{3}$$



Examples: Write a maxima program to evaluate following line integral with constant limits:

1. $\int_c (3x + y)dx + (2y - x)dy$ along the curve $y = x^2 + 1$ from $(0, 1)$ to $(3, 10)$.
2. $\int_c (2xy - 1)dx + (x^2 + 1)dy$ along the parabola $y = x + 1$ from $(0, 1)$ to $(2, 3)$.
3. $\int_c (x + y)dx + (y - x)dy$ along the parabola $y^2 = x$ from $(1, 1)$ to $(4, 2)$.

Question 2: Write a maxima program to evaluate line integral with variable limits $\int_c xy dx + x^2z dy + xyz dz$ along the curve $c: x = e^t, y = e^{-t}, z = t^2$ and $1 \leq t \leq 2$.

Program:

```
kill(all)$
depends([x,y,z],t)$
x:%e^t$
y:%e^(-t)$
z:t^2$
Integrand: x*y* diff(x,t)+x^2*z*diff(y,t)+x*y*z*diff(z,t)$
I:integrate(Integrand,t,1,2)$
print("Integrand =",Integrand)$
print("Integral value = ",I)$
```

Output:

```
"Integrand ="" "-t^2*%e^t+%e^t+2*t^3" "
```

```
"Integral value = "" "-(2*%e^2-15)/2" "
```

```
(%i8) kill(all)$
depends([x,y,z],t)$
x:%e^t$
y:%e^(-t)$
z:t^2$
Integrand: x·y·diff(x,t)+x^2·z·diff(y,t)+x·y·z·diff(z,t)$
l:integrate(Integrand,t,1,2)$
print("Integrand =",Integrand)$
print("Integral value = ",l)$
```

$$\text{Integrand} = -t^2 e^t + e^t + 2t^3$$

$$\text{Integral value} = -\frac{2e^2 - 15}{2}$$

Examples: Write a maxima program to evaluate following line integral with variable limits:

1. $\int_c (3x - 2y)dx + (y + 2z)dy - x^2 dz$ where c is the curve given by $x = t, y = 2t^2, z = 3t^3$ and $0 \leq t \leq 1$.
2. $\int_c (x^2 - y)dx + (y^2 + x)dy$ where c is the curve given by $x = t, y = t^2 + 1$ and $0 \leq t \leq 1$.
3. $\int_c x dx - y dy$ along the circle $c: x = a \cos t, y = 1 + \sin t$ and $-\frac{\pi}{2} \leq t \leq 0$.

Program 11

Program to evaluate the double integrals with constant and variable limits.

Question 1: Write a maxima program to evaluate double integral with constant limits $\int_0^1 \int_0^2 x^3 e^y dx dy$.

Program:

```
(%i4) kill(all)$
      Integrand:x^3*%e^(y)$
      I:integrate(integrate(Integrand,x,0,2),y,0,1)$
      print("Integrand =",Integrand)$
      print("Integral value = ",I)$
```

Output:

```
"Integrand =" "x^3*%e^y" "
"Integral value = " "4*(%e-1)" "
(%i4) kill(all)$
      Integrand:x^3*%e^(y)$
      I:integrate(integrate(Integrand,x,0,2),y,0,1)$
      print("Integrand =",Integrand)$
      print("Integral value = ",I)$

      Integrand = x3 %ey
      Integral value = 4 (%e-1)
```

Examples: Write a maxima program to evaluate following double integral with constant limits:

1. $\int_0^1 \int_0^2 (x + y) dx dy$
2. $\int_0^a \int_0^b (x^2 + y^2) dy dx$
3. $\int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{6}} \sin x \cos y dx dy$
4. $\int_0^1 \int_0^1 \frac{1}{(x+y+1)} dx dy$
5. $\int_3^4 \int_1^4 (xy + e^y) dx dy$

Question 2: Write a maxima program to evaluate double integral with variable limits $\int_1^4 \int_0^x xy \, dy \, dx$.

Program:

```
kill(all)$
Integrand:x*y$
I: integrate(integrate(Integrand,y,0,x),x,1,4)$
print("Integrand =",Integrand)$
print("Integral value = ",I)$
```

Output:

```
"Integrand =" "x*y" "
"Integral value = " "255/8" "
(%i4) kill(all)$
Integrand:x·y$
I: integrate(integrate(Integrand,y,0,x),x,1,4)$
print("Integrand =",Integrand)$
print("Integral value = ",I)$

Integral value = 255
                  8
```

Examples: Write a maxima program to evaluate following double integral with variable limits:

1. $\int_0^1 \int_0^{\sqrt{1+x^2}} \frac{dy \, dx}{1+x^2+y^2}$
2. $\int_0^2 \int_0^{x^2} x(x^2 + y^2) dy dx$
3. $\int_0^a \int_0^{\sqrt{a^2-x^2}} \sqrt{a^2 - x^2 - y^2} dy dx$
4. $\int_0^\pi \int_0^{\sin y} dx dy$
5. $\int_0^1 \int_0^{y^2} e^{\frac{x}{y}} dy dx$

Program 12

Program to evaluate the triple integrals with constant and variable limits.

Question 1: Write a maxima program to evaluate triple integral with constant limits $\int_1^2 \int_0^1 \int_{-1}^1 (x^2 + y^2 + z^2) dx dy dz$.

Program:

```
(%i4) kill(all)$
      Integrand:x^2+y^2+z^2$
      I:integrate(integrate(integrate(Integrand, x, -
      1,1),y,0,1),z,1,2)$
      print("Integrand =",Integrand)$
      print("Integral value = ",I)$
```

Output:

```
"Integrand =" "z^2+y^2+x^2" "
"Integral value = " "6" "

(%i4) kill(all)$
      Integrand:x^2+y^2+z^2$
      I:integrate(integrate(integrate(Integrand,x,-1,1),y,0,1),z,1,2)$
      print("Integrand =",Integrand)$
      print("Integral value = ",I)$

      Integrand = z^2 + y^2 + x^2
      Integral value = 6
```

Examples: Write a maxima program to evaluate following triple integral with constant limits:

1. $\int_1^2 \int_0^1 \int_0^1 e^{x+y+z} dx dy dz$
2. $\int_1^2 \int_0^2 \int_1^3 x^2 yz dx dy dz$
3. $\int_1^2 \int_{-2}^{-1} \int_2^3 \frac{1}{xyz} dx dy dz$
4. $\int_1^2 \int_1^2 \int_2^2 \left(\frac{x}{y} + \frac{y}{z} + \frac{z}{x} \right) dz dy dx$

Question 2: Write a maxima program to evaluate triple integral with variable limits $\int_{-1}^1 \int_0^z \int_{x-z}^{x+z} (x + y + z) dx dy dz$

Program:

```
kill(all)$
Integrand:x+y+z$
I:integrate(integrate(integrate(Integrand,x,x-z
,x+z),y,0,z),z,-1,1)$
print("Integrand =",Integrand)$
print("Integral value = ",I)$
```

Output:

"Integrand =" "z+y+x" "

"Integral value = "" "(8*x+3)/6-(4*x-9)/6" "

```
(%i4) kill(all)$
Integrand:x+y+z$
I:integrate(integrate(integrate(Integrand,x,x-z,x+z),y,0,z),z,-1,1)$
print("Integrand =",Integrand)$
print("Integral value = ",I)$

Integrand = z+y+x

Integral value =  $\frac{8x+9}{12} + \frac{8x-9}{12}$ 
```

Examples: Write a maxima program to evaluate following triple integral with variable limits:

$$1. \int_1^1 \int_0^1 \int_0^1 e^{x+y+z} dx dy dz$$

$$2. \int_0^a \int_0^{\sqrt{a^2-x^2}} \int_0^{\sqrt{a^2-x^2-y^2}} xyz dz dy dx$$



BLDEA'S S.B. ARTS AND K.C.P. SCIENCE

COLLEGE, VIJAYAPUR

(Affiliated to Rani Channamma University, Belagavi)

Practical's on DSC 3
ORDINARY DIFFERENTIAL EQUATION
AND REAL ANALYSIS-I

(AS PER NEP-2020)

MANUAL

B. Sc. III Semester

Name:.....

Class:.....

UUCMS No:.....

DEPARTMENT OF MATHEMATICS

Program I: Verification of Exact Differential Equation.**Maxima code to verify the exactness of the differential equation**

$$(x^2 - ay)dx - (ax - y^2)dy = 0.$$

```

kill(all)$
z:(x^2-a*y)·dx-(a·x-y^2)·dy$
m:coeff(z,dx)$
n:coeff(z,dy)$
print("M=",m)$
print("N=",n)$
my:ratsimp(diff(m,y))$
nx:ratsimp(diff(n,x))$
print("Partial derivative of m wrt y=",my)$
print("Partial derivative of n wrt x=",nx)$
ny:coeff(n,x,0)$
intm:integrate(m,x)$
intn:integrate(ny,y)$
if(my=nx) then
(
  disp("The given equation is exact"),
  zs:intm+intn,
  print("The solution is:"),
  disp(zs+c)
)
else
  disp("The given equation is not exact");

```


Output:

$$M = (x^2 - ay)$$

$$N = (ax - y^2)$$

Partial derivative of m wrt y= -a

Partial derivative of n wrt x= -a

The given equation is exact

The solution is:

$$\frac{y^3}{3} - axy + \frac{x^3}{3} + c$$

Example:

1. Solve $(3x^2 + 6xy^2)dx + (6xy^2 + 4y^3)dy = 0$
2. Solve $(a^2 - 2xy - y^2)dx + (x + y)^2dy = 0$
3. Solve $\sec^2 x \tan y dx + \sec^2 \tan x dy = 0$

Program II: Solution of differential equations that are solvable for x.**Maxima code to find the solution of the differential equation $x = y + p$.**

```
kill(all)$
x:y+p$
x(y):=y+p(y)$
d:diff(x(y),y)$
eq:1/p-d$
q:solve(eq,'diff(p(y),y,1))$
print(q)$
sol1:integrate(1,y)=-integrate((p/(p-1)),p)+c$
print("The parametric solution of the given problem is x=",x,"and",sol1)$
```

Output:

$$\left[\frac{d}{dy} p(y) = -\frac{p-1}{p} \right]$$

The parametric solution of the given problem is $x = y + p$ and $y = -p - \log(p - 1) + c$

Example:

1. Solve $y = 2px + y^2p^3$
2. Solve $y - 2px + yp^2 = 0$
3. Solve $y^2 \log y = xpy + p^2$

Program III: Solution of differential equations that are solvable for y.**Maxima code to find the solution of the differential equation $y = \sin p - p \cos p$**

```
kill(all)$
y:sin(p)-p*cos(p)$
y(x):=sin(p(x))-p(x)*cos(p(x))$
d:diff(y(x),x)$
q:p(x)-d$
solve(q,'diff(p(x),x,1))$
sol:integrate(1,x)=integrate(sin(p),p)+c$
print("The parametric solution y=",y,"and",sol)$
```

Output:

The parametric solution $y = \sin p - p \cos p$ and $x = c - \cos(p)$

Examples:

1. Solve $y = 2px + p^4x^2$
2. Solve $y = -px + x^4p^2$
3. Solve $y = 2px - p^2$

Program IV: Solution of differential equations that are solvable for p.

Maxima code to find the solution of the differential equation $x^2p^2 + xyp - 6y^2 = 0$

```
kill(all)$
equ:x^2·p^2+x·y·p-6·y^2$
solve(equ,p)$
r:subst(p='diff(y,x),%)$
r1:r[1]$
ode2(r1,y,x)$
sol1:(lhs(%)-rhs(%))$
r2:r[2]$
ode2(r2,y,x)$
sol2:(lhs(%)-rhs(%))$
print("The complete solution is:",sol1·sol2=0)$
```

Output:

The complete solution is: $\left(y - \frac{y^3}{x^3}\right)(y - y^3x^2) = 0$

Examples:

1. Solve $p^2 + p = 6$
2. Solve $p^2 - 5p - 6 = 0$
3. Solve $p^2 - 7p + 12 = 0$

Program V: To find the singular solution by using Clairaut's form.

Maxima code to find the solution of the differential equation $y = px - \frac{a}{p}$.

```
kill(all)$
A:y-p*x-a/p$
B:subst(p=c,A=0)$
print("The general solution is",B)$
C:diff(A,p)$
D:solve(C,p)$
a:2$
print("The singular solution is")$
for i:1 thru length(D) do(
E[i]:subst(p=rhs(D[i]),A=0),
disp(radcan(E[i])));
```

Output:

The general solution is $y - cx - \frac{a}{c} = 0$

The singular solution is

$$y + 2\sqrt{a}\sqrt{x} = 0$$

$$y - 2\sqrt{a}\sqrt{x} = 0$$

Examples:

1. Find the general solution of $p = \log(y - px)$
2. Find the general solution of $\sin p x \cos y - \cos p x \sin y = p$
3. Find the general solution of $(x^2 - 1)p^2 - 2xyp + y^2 - 1 = 0$

Program VI: Examining the Convergence of the Sequence.

Maxima code to check the convergence of $x_n = \left(1 + \frac{1}{n}\right)^n$.

```
kill(all)$
xn:(1+1/n)^n;
lim:limit(xn,n,inf);
if abs(lim)=inf then
print("sequence is divergent")
elseif abs(lim)#inf and abs(lim)#ind then
print("sequence is convergent")
else
print("Sequence is oscillatory")$
```

Output:

$$\left(1 + \frac{1}{n}\right)^n$$

%e

Sequence is convergent

Examples:

1. $\frac{2n+3}{3n+4}$
2. $1 - \frac{1}{n}$
3. $\frac{3n+4}{2n+1}$

Program VII :Verification of Exponential Series.

Maxima code to verify the exponential series : $\sum_{n=1}^{\infty} \left(\frac{n^3}{n!}\right)$

```
kill(all)$
```

```
load("simplify_sum")$
```

```
u(n):=n^3/factorial(n)$
```

```
S:sum(u(n),n,1,inf)$
```

```
print("The given series is:", S)$
```

```
s1:simplify_sum(S)$
```

```
print("Sum of the series",s1)$
```

Output:

The given series is: $\sum_{n=1}^{\infty} \left(\frac{n^3}{n!}\right)$

Sum of the series is 5%e

Examples:

$$1. \sum_{n=1}^{\infty} \frac{2^{n-1}}{(n+1)!}$$

$$2. \sum_{n=1}^{\infty} \frac{n^2}{n!}$$

$$3. \sum_{n=1}^{\infty} \frac{n(n+1)}{2n!}$$

Program VIII: Verification of Logarithmic Series.

Maxima code to verify the logarithmic series $\sum_{n=1}^{\infty} \frac{(-1)^{n+1} x^n}{n}$

```
kill(all)$
```

```
u(n):=((-1)^(n+1)·(x)^n/n)$
```

```
s:sum(u(n),n,1,inf);
```

```
load("simplify_sum")$
```

```
sum:simplify_sum(s)$
```

```
print("Sum of the series",sum)$
```

Output:

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1} x^n}{n}$$

Sum of the series is: $\log(x + 1)$

Examples:

1. $\sum_{n=1}^{\infty} \frac{x^n}{n}$

2. $\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}$

3. $\sum_{n=1}^{\infty} \frac{1}{n} - \frac{2}{2n+1}$

Program IX: De'Alembert's Ratio Test

Maxima code to check the convergence of the series $\sum_{n=1}^{\infty} \frac{1}{n^2}$.

```
kill(all)$
u(n):=1/n^2;
D:limit(u(n+1)/u(n),n,inf);
if D<1 then
disp("By ratio test the given series is convergent")
else
if D>1 then
disp("By ratio test the given series is divergent")
else
disp("Ratio test fails")$
```

Output:

$$u(n) := \frac{1}{n^2}$$

1

Ratio test fails

Examples:

1. $\sum_{n=1}^{\infty} \frac{n!}{n}$

2. $\sum_{n=1}^{\infty} \frac{n^2}{3^n}$

3. $\sum_{n=1}^{\infty} \sqrt{\frac{n+1}{n+2}}$

Program X: Cauchy's Root Test :

Maxima code to check the convergence of the series $\sum_{n=1}^{\infty} \frac{n^2}{(n+1)^n 2^n}$

```
kill(all)$
u(n):=(n/(2*(n+1)))^n$
s:sum(u(n),n,1,inf);
disp("The given series is:",s)$
C:limit((u(n))^(1/n),n,inf);
if C<1 then
disp("By cauchy's root test series is convergent")
else
if C>1 then
disp("By cauchy's root test series is divergent")
else
disp("Cauchy's test fails")$
```

Output:

The given series is:

$$\sum_{n=1}^{\infty} \frac{n^2}{(n+1)^n 2^n}$$

$$\frac{1}{2}$$

By cauchy's root test series is convergent

Example:

$$1. \sum_{n=1}^{\infty} \left(1 + \frac{1}{n}\right)^{n^2}$$

$$2. \sum_{n=1}^{\infty} \left[\left(\frac{n+1}{n} \right) - \left(\frac{n+1}{n} \right)^{n+1} \right]^n$$

$$3. \sum_{n=1}^{\infty} \frac{n^{n+1}}{(n+1)^n}$$

Department of Mathematics (NEP)

Program XI: Convergence of the Alternating Series Using Leibnitz's Theorem.

Maxima code to check the convergence of the series $\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{\sqrt{n-1}}$

```
kill(all)$
```

```
u(n):=1/sqrt(n-1)$
```

```
s:sum((((-1)^(n-1))·u(n),n,1,inf)$
```

```
print("The given series is:",s)$
```

```
L:limit(u(n),n,inf);
```

```
if L=0 and u(101)<u(100)then
```

```
print("By Leibnitz's test series is convergent")
```

```
else
```

```
print("By Leibnitz's test series is not convergent")$
```

Output:

The given series is: $\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{\sqrt{n-1}}$

0

By Leibnitz's test series is convergent

Example:

$$1. \sum_{n=1}^{\infty} (-1)^{n+1} \frac{n}{2n-1}$$

$$2. \sum_{n=1}^{\infty} (-1)^n \frac{1}{n}$$

$$3. \sum_{n=1}^{\infty} (-1)^{n+1} \frac{1}{n!}$$

Program XII: Verification of Cauchy's Integral Test.

Maxima code to check the convergence of the series $Un = \sum_{n=1}^{\infty} \frac{1}{n^2+1}$ by using cauchy's integral test .

```
kill(all)$

print("Cauchy's integral test")$

U(n):=1/(n^2+1)$

ss:sum(U(n),n,1,inf)$

print("U(n)=",ss)$

f(x):=1/(x^2+1)$

d(x):=ratsimp(f(x)-f(x+1))$

print("f(x)-(x+1):",d(x))$

if d(1)>0 then

(

print("The sequence f(x) is monotonically decreasing"),

print("Cauchy's integral test ids applicable"),

s:integrate(f(x),x,1,inf),

ifs#inf then

(

print("f(x) is convergent"),

print("Hence by Cauchy's integral test U(n) is convergent")

)

else

print("U(n) is divergent")

)
```

else

print("Cauchy's integral test cannot be applied")\$

Output:

Cauchy's integral test

$$U_n = \sum_{n=1}^{\infty} \frac{1}{n^2 + 1}$$

$$f(x) - f(x+1): \frac{2x+1}{x^4 + 2x^3 + 3x^2 + 2x + 2}$$

The sequence $f(x)$ is monotonically decreasing

Cauchy's integral test is applicable

$f(x)$ is convergent

Hence by Cauchy's integral test $U(n)$ is convergent

Examples:

$$1. \sum_{n=1}^{\infty} \left[\frac{\log n}{\log(n+1)} \right]^{n^2 \log n}$$

$$2. \sum_{n=1}^{\infty} \left(\frac{n+1}{n+2} \right)^n x^n \text{ where } (x > 0)$$



BLDEA'S S.B. ARTS AND K.C.P. SCIENCE

COLLEGE, VIJAYAPUR

(Affiliated to Rani Channamma University, Belagavi)

Practical's on DSC 4
PARTIAL DIFFERENTIAL EQUATIONS
AND INTEGRAL TRANSFORMS

(AS PER NEP-2020)

MANUAL

B. Sc. IV Semester

Name:.....

Class:.....

UUCMS No:.....

DEPARTMENT OF MATHEMATICS

Program I : Solution for Linear Partial Differential equation of type I to type IV.

Solve $pq=1$

/*Type I- PDE Equations of the form $f(p,q)=0$ */

kill(all)\$

print("The given equation is")\$

eqn:($p*q=1$)\$

disp(eqn)\$

z: $a*x+b*y+c$ \$

print("Substitute $p=a$ and $q=b$ in given equation")\$

h:subst([$p=a,q=b$],eqn)\$

print("We get",h)\$

disp("solving for a and b")\$

solve(h,a);

h1:subst(%,z)\$

disp("The required solution is:",h1)\$

Output:

The given equation is

$$pq = 1$$

Substitute $p=a$ and $q=b$ in given equation

$$\text{We get } ab = 1$$

solving for a and b

$$\left[a = \frac{1}{b} \right]$$

The required solution is :

$$by + \frac{x}{b} + c$$

Examples:

$$1. \quad p^2 + q^2 = 1$$

$$2. \quad \sqrt{p} + \sqrt{q} = 1$$

$$3. \quad p = e^q$$

Solve $p(1+q)=qz$

/*TypeII-PDE Equations of the form $f(p,q,z)=0$ */

kill(all)\$

print("The given equation is")\$

eqn:p*(1+q)=q*z\$

disp(eqn)\$

disp("substitute $p=dz/du$ and $q=a.dz/du$ ")\$

eqn1:subst([p='diff(z,u),q=a*'diff(z,u)],eqn)\$

disp(eqn1)\$

disp("solving for dz/du and substitute $u=ax+y$ ")\$

h1:solve(eqn1,'diff(z,u))\$

disp(h1)\$

ode2(h1[1],z,u)\$

h2:subst(u=x+a*y,%)\$

disp("solution is:",h2)\$

Output:

The given equation is

$$p(q+1) = qz$$

$$\text{substitute } p=dz/du \text{ and } q=a.dz/du \quad \frac{d}{du}z \left(a \left(\frac{d}{du}z \right) + 1 \right) = a z \left(\frac{d}{du}z \right)$$

Solving for dz/du and substitute $u = ax + y$

$$\left[\frac{d}{dz}z = \frac{az-1}{a}, \frac{d}{dz}z = 0 \right]$$

solution is :

$$z = \left(\frac{{}^{\%}e^{-ay-x}}{a} + {}^{\%}c \right) {}^{\%}e^{ay+x}$$

Examples:

$$1. \quad p(1-q^2) = q(1-z)$$

$$2. \quad pq = z$$

$$3. \quad p^3 + q^3 = 3pqz$$

Solve $p+x=q+y$

```
/*TypeIII-PDE Equations of the form  $f_1(p,x) = f_2(q,y)$  */
```

```
kill(all)$
```

```
print("The given PDE is")$
```

```
eqn:p+x=q+y$
```

```
disp(eqn)$
```

```
d:(p+q*'diff(y,x))$
```

```
r1:lhs(eqn)=k$
```

```
disp(r1)$
```

```
r2:rhs(eqn)=k$
```

```
disp(r2)$
```

```
print("solving for p from LHS")$
```

```
h1:solve(r1,p)$
```

```
disp(h1)$
```

```
print("solving for q from RHS")$
```

```
h2:solve(r2,q)$
```

```
disp(h2)$
```

```
print("substituting p and q in  $dz=pdx+qdy$  and integrate")$
```

```
subst(h1,d)$
```

```
A:subst(h2,%)$
```

```
ode2((A),y,x)$
```

```
z: rhs(%)-lhs(%)$
```

```
disp("the solution is  $z=$ ",z)$
```

Output:

The given PDE is

$$x + p = y + q$$

$$x + p = K$$

$$y + q = k$$

solving for p from LHS

$$[p = K - x]$$

solving for q from RHS

$$[q = k - y]$$

substituting p and q in $d=px+qdy$ and integrate

the solution is $z=$

$$\frac{y^2 - 2ky}{2} + \frac{x^2 - 2Kx}{2} + \%c$$

Examples:

$$1. \quad p^2 - x = q^2 + y$$

$$2. \quad \sqrt{p} + \sqrt{q} = x + y$$

$$3. \quad pe^y = qe^x$$

Solve $z = px + qy + \sqrt{\frac{pq}{p+q}}$

/*TypeIV-PDE Equations of the form $z=px+qy+f(p,q)$ Clariaut's equation*/

kill(all)\$

print("The given PDE is")\$

eqn: $z=p*x+q*y+sqrt((p*q)/(p+q))$ \$

disp(eqn)\$

disp("substitute $p=a$ & $q=b$ in given equation")\$

soln:subst([$p=a, q=b$],eqn)\$

disp("solution is s:", soln)\$

Output:

The given PDE is

$$z = qy + px + \sqrt{\frac{pq}{q+p}}$$

substitute $p=a$ and $q=b$ in given equation

solution is s:

$$z = by + ax + \sqrt{\frac{ab}{b+a}}$$

Examples:

1. $z = px + qy + \log(pq)$

2. $z = px + qy + (pq)$

3. $z = px + qy + p^2 + q^2$

Program II : Solution of second order homogenous partial differential equation with constant coefficient.

Solve $D^2 + 3DD^1 + 2(D^1)^2 = 0$

/* second order linear PDE with constant co-efficients */

kill(all)\$

F(D,D1)*z=0\$

disp("The given PDE is")\$

F(D,D1):=D^2+3*D*(D1)+2*(D1)^2=0\$

disp(F(D,D1))\$

print("substitute D=m & D1=1")\$

ae:F(m,1)\$

print("Auxiliary equation is",ae)\$

k:allroots(ae)\$

print("roots are",k)\$

k1:rhs(k[1])\$

k2:rhs(k[2])\$

a1:f(y+k1*x)+g(y+k2*x)\$

a2:f(y+k1*x)+x*g(y+k2*x)\$

if k1#k2 then

disp("solution is",z=a1)

elsedisp("solution is ",z=a2)\$

Output:

The given PDE is:

$$2D1^2 + 3DD1 + D^2 = 0$$

substitute $D=m$ & $D1=1$

auxiliary equation is $m^2 + 3m + 2 = 0$

roots are $[m = -1.0, m = -2.0]$

solution is

$$z = f(y - 1.0x) + g(y - 2.0x)$$

Examples:

$$1. D^2 - DD' - 6D'^2 = 0$$

$$2. D^2 + 6D^2D' + 9D'^2 = 0$$

Solve $D^2 - DD' - 2(D')^2 = e^{(x+2y)}$

```
kill(all)$
```

```
ratprint:false$
```

```
F(D,D1)*z=f(x,y)$
```

```
disp("The given PDE is")$
```

```
F(D,D1):=D^2-D*D1-2*D1^2$
```

```
disp(F(D,D1))$
```

```
print("substitute D=m & D1=1")$
```

```
ae:F(m,1)$
```

```
print("Auxiliary equation is",ae)$
```

```
h:allroots(ae=0)$
```

```
print("roots are",h)$
```

```
h1:rhs(h[1])$
```

```
h2:rhs(h[2])$
```

```
cf1:f1(y+h1*x)+g1(y+h2*x)$
```

```
cf2:f1(y+h1*x)+x*g1(y+h2*x)$
```

```
if h1#h2 then
```

```
(CF:cf1)
```

```

else
(CF:cf2)$
print("Complimentary function is CF=",CF)$
/*For particular integral*/
f(x,y):=%e^(x+2*y)$
print("RHS=",f(x,y))$
I1:integrate(f(x,c-h1*x),x)$
f3(x,y):=ratsimp(subst([c=y+h1*x],I1))$
integrate(f3(x,c-h2*x),x)$
PI:ratsimp(subst([c=y+h2*x],%))$
print("Particular Integral is",PI)$
z:CF+PI$
print("Solution is",z)$

```

Output:

The given PDE is

$$-2D_1^2 - DD_1 + D^2 = 0$$

substitute $D=m$ and $D_1=1$

auxiliary equation is $m^2 - m - 2 = 0$

roots are $[m = -1.0, m = 2.0]$

complimentary function is $CF = g_1(y+2.0x) + f_1(y-1.0x)$

$RHS = e^{2y+x}$

Particular integral is $-\frac{e^{2y+x}}{9}$

Solution is $g_1(y+2.0x) + f_1(y-1.0x) - \frac{e^{2y+x}}{9}$

Example:

$$1. D^2 - DD' + 6D'^2 = e^{(x+y)}$$

$$2. D^2 - DD' + 6D'^2 = e^{(x-y)}$$

Program III: Laplace Transforms of some common functions:

$$1. L[a] = \frac{a}{s} \quad 2. L[e^{at}] = \frac{1}{s-a} \quad 3. L[e^{-at}] = \frac{1}{s+a} \quad 4. L[\cosh at] = \frac{s}{s^2 - a^2} \quad 5. L[\sinh at] = \frac{a}{s^2 - a^2}$$

$$6. L[\cos at] = \frac{s}{s^2 + a^2} \quad 7. L[\sin at] = \frac{a}{s^2 + a^2} \quad 8. L[t^n] = \frac{n!}{s^{n+1}}, n \in \mathbb{N} \quad 9. L[t^n] = \frac{\Gamma(n+1)}{s^{n+1}}, n \text{ is non integer}$$

$$10. L[f^{(n)}(t)] = s^n L[f(t)] - s^{n-1} f(0) - s^{n-2} f'(0) - \dots - f^{(n-1)}(0) \quad 11. L[t^n f(t)] = (-1)^n \frac{d^n}{ds^n} F(s)$$

Evaluate Laplace transform of $f(t) = e^{at}$

/*Laplace Transform of $e^{(at)}$ */

kill(all)\$

disp("The given function is")\$

f:%e^(a*t)\$

print("f(t)=",f)\$

L:laplace(f,t,s)\$

disp("Laplace transform of given function is ")\$

print("L[f(t)=",L)\$

Output:

The given function is

$$f(t) = e^{at}$$

Laplace transform of given function is

$$L[f(t)] = \frac{1}{s-a}$$

Evaluate Laplace transform of $f(t) = \sin^2 at$


```
kill(all)$  
  
disp("The given function is")$  
  
f:(sin(a*t))^2$  
  
print("f(t)=",f)$  
  
L:laplace(f,t,s)$  
  
disp("Laplace transform of given function is ")$  
  
print("L[f(t)=",L)$
```

Output:

The given function is

$$f(t) = \sin(at)^2$$

Laplace transform of given function is

$$L[f(t)] = \frac{2a^2}{s^3 + 4a^2s}$$

Program IV: Laplace Transforms of periodic functions:

Find the laplace transform of $f(t)=t^2$, $0<t<2$ & $f(t+2)=f(t)$

```
kill(all)$
disp("The given function is")$
f:t^2$
print("f(t)=",f)$
print("Enter the period")$
t1:2$
print("T=",t1)$
a:1/(1-%e^(-t1*s))$
I:integrate(%e^(-s*t)*f,t,0,2)$
L:(a*I)$
print("Laplace transform of given periodic function",L)$
```

Output:

The given function is

$$f(t) = t^2$$

Enter the period

$$T = 2$$

Laplace transform of given periodic function $\frac{\frac{2}{s^3} \frac{(4s^2+4s+2)e^{-2s}}{s^3}}{1-e^{-2s}}$

Example:

1. $f(t) = e^{-t}$, $0 < t < 1$
2. $f(t) = e^t$, $0 < t < 2$

Find Laplace transform of periodic function $f(t) = \begin{cases} 3t & 0 < t < 2 \\ 6 & 2 < t < 4 \end{cases}$

```
kill(all)$
disp("The given function is")$
f1:3*t$
f2:6$
disp(f1,f2)$
print("Enter the period")$
t1:4$
print("T=",t1)$
a:1/(1-%e^(-t1*s))$
I1:integrate(%e^(-s*t)*f1,t,0,2)$
I2:integrate(%e^(-s*t)*f2,t,2,4)$
I:I1+I2$
L:ratsimp(a*I)$
print("Laplace transform of given periodic functionis",L)$
```

Output:

The given function is

3t

6

Enter the period

Laplace transform of given periodic function is $\frac{3e^{4s}-3e^{2s}-6s}{s^2e^{4s}-s^2}$

Example:

$$1. f(t) = \begin{cases} E, & 0 \leq t \leq \frac{T}{2} \\ -E, & \frac{T}{2} \leq t \leq T \end{cases} \text{ with } f(t+T) = f(t)$$

$$2. f(t) = \begin{cases} 1, & 0 \leq t \leq \frac{a}{2} \\ -1, & \frac{a}{2} \leq t \leq a \end{cases} \text{ with } f(t+a) = f(t)$$

Program V : Inverse Laplace transform:

1. $L^{-1}\left[\frac{1}{s}\right] = 1$
2. $L^{-1}\left[\frac{a}{s}\right] = a$
3. $L^{-1}\left[\frac{1}{s-a}\right] = e^{at}$
4. $L^{-1}\left[\frac{1}{s+a}\right] = e^{-at}$
5. $L^{-1}\left[\frac{1}{s^2-a^2}\right] = \frac{1}{a}\sinh at$
6. $L^{-1}\left[\frac{s}{s^2-a^2}\right] = \cosh at$
7. $L^{-1}\left[\frac{1}{s^2+a^2}\right] = \frac{1}{a}\sin at$
8. $L^{-1}\left[\frac{s}{s^2+a^2}\right] = \cos at$
9. $L^{-1}\left[\frac{1}{s^{n+1}}\right] = \frac{t^n}{n!}, n=0,1,2,\dots$
10. $L^{-1}\left[\frac{1}{s^{n+1}}\right] = \frac{t^n}{\Gamma(n+1)}, n=0,1,2,\dots \text{ or non integer } < 0$

Evaluate inverse laplace transform of $L^{-1}\left[\frac{1}{s-2}\right]$

/*Inverse Laplace Transform*/

kill(all)\$

disp("The given function is")\$

f:1/(s-2)\$

print("f(s)=",f)\$

L:ilt(f,s,t)\$

disp("Inverse laplace transform of given function is ")\$

print("[f(t)=",L)\$

Output:

The given function is

$$f(s) = \frac{1}{s-2}$$

Inverse laplace transform of given function is

$$[f(t)] = e^{2t}$$

Evaluate inverse laplace transform of $L^{-1}\left[\frac{1}{s(s-1)(s-2)}\right]$

```
kill(all)$
```

```
disp("The given function is")$
```

```
f:1/(s*(s-1)*(s-2))$
```

```
print("f(s)=",f)$
```

```
L:ilt(f,s,t)$
```

```
disp("Inverse laplace transform of given function is ")$
```

```
print("[f(t)=",L)$
```

Output:

The given function is

$$f(s) = \frac{1}{(s-2)(s-1)s}$$

Inverse laplace transform of given function is

$$[f(t) =] \frac{e^{2t}}{2} - e^t + \frac{1}{2}$$

Program VI: Verificaton of convolution theorem**Verify convolution theorem for the functions $f(t) = e^t$ & $g(t) = \cos t$**

/*Convolution theorem*/

kill(all)\$

assume(t>0)\$

print("Given functions are")\$

f(t):=%e^t\$

g(t):=cos(t)\$

print("f(t)=",f(t))\$

print("g(t)=",g(t))\$

L1:laplace(f(t),t,s)\$

print("L[f(t)=",L1)\$

L2:laplace(g(t),t,s)\$

print("L[g(t)=",L2)\$

I1:integrate(f(t-u)*g(u),u,0,t)\$

LHS:ratsimp(laplace(I1,t,s))\$

print("LHS=",LHS)\$

RHS:ratsimp(L1*L2)\$

print("RHS=",RHS)\$

if LHS=RHS then

print("Convolution theorem satisfied")

else

print("Convolution theorem is not satisfied")\$

Output:

Given functions are

$$f(t) = e^t$$

$$g(t) = \cos(t)$$

$$L[f(t)] = \frac{1}{s-1}$$

$$L[f(t)] = \frac{s}{s^2 + 1}$$

$$\text{LHS} = \frac{s}{s^3 - s^2 + s - 1}$$

$$\text{RHS} = \frac{s}{s^3 - s^2 + s - 1}$$

Convolution theorem satisfied

Examples:

1. $f(t) = 1$ and $g(t) = \sin(t)$.
2. $f(t) = \cos(t)$ and $g(t) = \sin(t)$.

Program VII: Solution for linear differential equation using laplace transform.

Solve $\frac{dy}{dx} + 5y = e^{-5x}$ using laplace transform at $y(0)=1$.

`/.Solution for linear ODE using Laplace transform/`

`kill(all)$`

`eq:'diff(y(x),x)+5*y(x)-%e^(-5*x)$`

`print("The given differential equation is",eq)$`

`laplace(%,x,s);`

`linsolve(%,laplace(y(x),x,s));`

`soln:subst(1,y(0),%)$`

`L:partfrac(rhs(soln[1]),s)$`

`print("Laplace transform of given equation is L[y(x)=",L)$`

`y:ilt(%,s,x)$`

`print("Required solution is y(x)=",y)$`

Output:

The given differential equation is $\frac{d}{dx} y(x) + 5 y(x) - e^{-5x}$

$\text{slaplace}(y(x), x, s) + 5 \text{laplace}(y(x), x, s) - \frac{1}{s+5} - y(0)$

$$[\text{laplace}(y(x), x, s) = \frac{y(0)s + 5y(0) + 1}{s^2 + 10s + 25}]$$

laplace transform of given equation is $L[y(x)] = \frac{1}{s+5} + \frac{1}{(s+5)^2}$

Required solution is $y(x) = x e^{-5x} + e^{-5x}$

Examples:

1. $y' - 5y = 0$ given that $y(0) = 2$.
2. $y' - 5y = e^{5x}$ given that $y(0) = 2$.

Solve $\frac{d^2 y}{dx^2} + 9y = 0$ with $y(0) = 0$ & $y'(0) = 2$ **using laplace transform at $y(0)=1$.**

```
kill(all)$
```

```
eq:'diff(y(x),x,2)+9*y(x)=0$
```

```
print("The given differential equation is",eq)$
```

```
laplace(%,x,s);
```

```
linsolve(%,laplace(y(x),x,s));
```

```
soln:subst(0,y(0),%)$
```

```
soln1:subst(2,'at('diff(y(x),x),x=0),%)$
```

```
L:partfrac(rhs(soln1[1]),s)$
```

```
print("Laplace transform of given equation is L[y(x)]=",L)$
```

```
y:ilt(%,s,x)$
```

```
print("Required solution is y(x)=",y)$
```

Output:

The given differential equation is $\frac{d^2}{dx^2} y(x) + 9 y(x) = 0$

$-\frac{d}{dx} y(x)|_{x=0} + s^2 \text{laplace}(y(x), x, s) + 9 \text{laplace}(y(x), x, s) - y(0) = 0$

$[\text{laplace}(y(x), x, s) = \frac{\frac{d}{dx} y(x)|_{x=0} + y(0)s}{s^2 + 9}]$

laplace transform of given equation is $L[y(x)] = \frac{2}{s^2 + 9}$

Required solution is $y(x) = \frac{2 \sin(3x)}{3}$

Examples:

1. $9y'' - 6y' + y = 0$ given $y(0) = 3$ and $y'(0) = 1$

Program VIII: Solution for Integral equation using laplace transform.

Find Laplace Transform of $\int_0^t \frac{(\cos at - \cos bt)}{t} dt$

```
kill(all)$
```

```
f:(cos(a*t)-cos(b*t))/t$
```

```
disp("Given integral equation is")$
```

```
f1:'integrate(f,t,0,t)$
```

```
print("f(t)=",f1)$
```

```
L:laplace(f,t,s)$
```

```
Lt:ratsimp((1/s)*L)$
```

```
disp("Laplace transform of given function is ")$
```

```
print("L[f(t)=",Lt)$
```

Output:

Given integral equation is

$$f(t) = \int_0^t \frac{\cos at - \cos bt}{t} dt$$

Laplace transform of given function is

$$L[f(t)] = \frac{\log(s^2 + b^2) - \log(s^2 + a^2)}{2s}$$

Examples:

$$1. f(t) = \int_0^t \frac{\cos 6t - \cos 4t}{t} dt$$

$$2. f(t) = \int_0^t \sin t dt$$

Program IX : Evaluation of Fourier series for simple functions.

Evaluate fourier series for $f(x) = x^2$ in $[-\pi, \pi]$

```
/·Fourier Series ·/
```

```
kill(all)$
```

```
load(fourie)$
```

```
print("The given function is")$
```

```
f(x):=x^2$
```

```
print("f(x)=",f(x))$
```

```
c:totalfourier(f(x),x,%pi)$
```

```
print("required fourier series is",c)$
```

OR

```
/·Fourier Series ·/
```

```
kill(all)$
```

```
load(fourie)$
```

```
print("The given function is")$
```

```
f(x):=x^2$
```

```
print("f(x)=",f(x))$
```

```
fourier(f(x),x,%pi)$
```

```
clist:foursimp(%)$
```

```
soln:fourexpend(clist,x,%pi,inf)$
```

```
print("Required fourier series is",soln)$
```

Output:

The given function is

$$f(x) = x^2$$

$$a_0 = \frac{\pi^2}{3}$$

$$a_n = \frac{2 \left(\frac{\pi^2 \sin(\pi n)}{n} - \frac{2 \sin(\pi n)}{n^3} + \frac{2 \pi \cos(\pi n)}{n^2} \right)}{\pi}$$

$$b_n = 0$$

$$a_0 = \frac{\pi^2}{3}$$

$$a_n = \frac{4(-1)^n}{n^2}$$

$$b_n = 0$$

Required fourier series is $4 \left(\sum_{n=1}^{\infty} \left(\frac{(-1)^n \cos(nx)}{n^2} \right) \right) + \frac{\pi^2}{3}$

Evaluate fourier series for $f(x) = x+x^2$ in $[-1, 1]$

```
kill(all)$
```

```
load(fourie)$
```

```
print("The given function is")$
```

```
f(x):=x+x^2$
```

```
print("f(x)=",f(x))$
```

```
c:totalfourier(f(x),x,1)$
```

```
print("required fourier series is",c)$
```

OR

```

kill(all)$

load(fourie)$

print("The given function is")$

f(x):=x+x^2$

print("f(x)=",f(x))$

fourier(f(x),x,1)$

clist:foursimp(%)$

soln:fourexpend(clist,x,1,inf)$

print("required fourier series is",soln)$

/·to plot the gaph·/

soln2:fourexpend(clist,x,1,100)$

wxplot2d([f(x),soln2], [x,-10,10])$

```

Output:

The given function is

$$f(x) = x^2 + x$$

$$a_0 = \frac{1}{3}$$

$$a_n = \frac{2 \sin(\pi n)}{n\pi} - \frac{4 \sin(\pi n)}{\pi^3 n^3} + \frac{4 \cos(\pi n)}{\pi^2 n^2}$$

$$b_n = \frac{2 \sin(\pi n)}{\pi^2 n^2} - \frac{2 \cos(\pi n)}{\pi n}$$

$$a_0 = \frac{1}{3}$$

$$a_n = \frac{4(-1)^n}{\pi^2 n^2}$$

$$b_n = \frac{2(-1)^n}{\pi n}$$

Required fourier series is $\frac{2 \sum_{n=1}^{\infty} \left(\frac{(-1)^n \sin(nx)}{n} \right)}{\pi} + \frac{4 \left(\sum_{n=1}^{\infty} \left(\frac{(-1)^n \cos(\pi nx)}{n^2} \right) \right)}{\pi^2} + \frac{1}{3}$

Program X: Evaluation half range sine and cosine series.

Evaluate half range fourier sine series for $f(x) = x^2$ in $[0, \pi]$

/·Half range sine series·/

kill(all)\$

load(fourie)\$

print("The given function is")\$

f(x):=x^2\$

print("f(x)=",f(x))\$

foursin(f(x),x,%pi)\$

clist:foursimp(%)\$

soln:fourexpend(clist,x,%pi,inf)\$

print("The half range sine series is",soln)\$

Output:

The given function is

$$f(x) = x^2$$

$$b_n = \frac{2 \left(\frac{\pi \sin(\pi n)}{n^2} - \frac{\pi^2 \cos(\pi n)}{n} + \frac{2 \cos(\pi n)}{n^3} - \frac{2}{n^3} \right)}{\pi}$$

$$b_n = - \frac{2(\pi^2 n^2 (-1)^n - 2(-1)^n + 2)}{\pi n^3}$$

The half range sine series is $-\frac{2 \sum_{n=1}^{\infty} \left(\frac{\pi^2 n^2 (-1)^n - 2(-1)^n + 2 \sin(nx)}{n} \right)}{\pi}$

Examples:

$$1. f(x) = 2x - 1 \text{ in } [0, \pi]$$

$$2. f(x) = x \text{ in } [0, \pi]$$

Evaluate half range fourier cosine series for $f(x) = x^2$ in $[0, \pi]$

/·Half range cosine series·/

```
kill(all)$
load(fourie)$
print("The given function is")$
f(x):=x^2$
print("f(x)=",f(x))$
fourcos(f(x),x,%pi)$
clist:foursimp(%)$
soln:fourexpand(clist,x,%pi,inf)$
print("The half range cosine series is",soln)$
```

Output:

The given function is

$$f(x) = x^2$$

$$a_0 = \frac{\pi^2}{3}$$

$$a_n = \frac{2 \left(\frac{\pi^2 \sin(\pi n)}{n} - \frac{2 \sin(\pi n)}{n^3} + \frac{2 \pi \cos(\pi n)}{n^2} \right)}{\pi}$$

$$a_0 = \frac{\pi^2}{3}$$

$$a_n = \frac{4(-1)^n}{n^2}$$

The half range cosine series is $4 \left(\sum_{n=1}^{\infty} \left(\frac{(-1)^n \cos(nx)}{n^2} \right) \right) + \frac{\pi^2}{3}$

Examples:

1. $f(x) = 2x - 1$ in $[0, \pi]$
2. $f(x) = x$ in $[0, \pi]$

ProgramXI: Evaluation of Fourier cosine transform

Evaluate Fourier cosine transform of $f(x)=x$ in $[0,\pi]$

```
/*Fourier cosine transform*/
```

```
kill(all)$
```

```
print("The given function is")$
```

```
f(x):=x$
```

```
print("f(x)=",f(x))$
```

```
a:0$
```

```
p:%pi$
```

```
s:integrate(f(x)*cos((n*x*%pi)/p),x,0,p)$
```

```
fs:ratsimp(s)$
```

```
print("Fourier Cosine Transform of given function is",fs)$
```

Output:

The given function is

$$f(x) = x$$

Fourier Cosine Transform of given function is $\frac{\pi n \sin(\pi n) + \cos(\pi n) - 1}{n^2}$

Examples:

1. $f(x) = x^2$ in $[0, \pi]$
2. $f(x) = \pi - x$ in $[0, \pi]$

ProgramXII: Evaluation of Fourier sine transform

Evaluate Fourier sine transform of $f(x)=3x$ in $[0,6]$

```
/*Fourier sine transform*/
```

```
kill(all)$
```

```
print("The given function is")$
```

```
f(x):=3*x$
```

```
print("f(x)=",f(x))$
```

```
a:0$
```

```
p:6$
```

```
s:integrate(f(x)*sin((n*x*%pi)/p),x,0,p)$
```

```
fs:ratsimp(s)$
```

```
print("Fourier Sine Transform of given function is",fs)$
```

Output:

The given function is

$$f(x) = 3x$$

Fourier Sine Transform of given function is $\frac{108 \sin(\pi n) - 108 \pi n \cos(\pi n)}{\pi^2 n^2}$

Examples:

1. $f(x) = x^2$ in $[0, 4]$
2. $f(x) = \pi - x$ in $[0, \pi]$

Department of Mathematics(NEP)



BLDEA'S S.B. ARTS AND K.C.P. SCIENCE

COLLEGE, VIJAYAPUR

(Affiliated to Rani Channamma University, Belagavi)

Practical's on DSC I
REAL ANALYSIS II AND COMPLEX
ANALYSIS
(AS PER NEP-2020)
MANUAL

B. Sc. V Semester

Name:.....

Class:.....

UUCMS No:.....

DEPARTMENT OF MATHEMATICS

CONTENTS

Sl. No.	Title	Page No
1	Contents	2
2	List of Experiments	3
3	Experiment1 Program on verification of Cauchy– Riemann Equations(Cartesian Form) or test for Analyticity.	4
4	Experiment2 Program on verification of Cauchy– Riemann Equations(Polar Form) or test for Analyticity.	14
5	Experiment3 Program to check whether a function is harmonic or not.	21
6	Experiment4 Program to construct analytic functions (Milne-Thomson Method).	27
7	Experiment5 Program to find cross-ratio of points and related concepts.	35
8	Experiment6 Program to find fixed points of bilinear transformations.	44
9	Experiment7 Program to verify De-Moivre’s theorem.	50
10	Experiment8 Program to check whether a given set of real numbers attains supremum or infimum.	
11	Experiment9 Program to find upper and the lower Riemann sums with respect to a given partition.	56
12	Experiment10 Program to test Riemann integrability.	57
13	Experiment11 Program to evaluate Riemann integral as a limit of sum.	59
14	Experiment 12 Program to evaluate $\Gamma(n)$ for n is integer and non-integer.	60
15	Experiment 13 Program to evaluate $B(m, n)$ for $m > 0$ and $n > 0$.	61

List of Experiments

For Fifth Semester DSC Mathematics
(Practical on Real Analysis-II and Complex Analysis)

(4 Hours per Week and 56 hours per Semester)

Elements of Partial differential equations and Integral transforms using FOSS.

1. Program on verification of Cauchy – Riemann Equations (Cartesian Form) or test for Analyticity.
2. Program on verification of Cauchy – Riemann Equations (Polar Form) or test for Analyticity.
3. Program to check whether a function is harmonic or not.
4. Program to construct analytic functions (Milne-Thomson Method).
5. Program to find cross-ratio of points and related concepts.
6. Program to find fixed points of bilinear transformations.
7. Program to verify De-Moivre's theorem.
8. Program to check whether a given set of real numbers attains supremum or infimum.
9. Program to find upper and the lower Riemann sums with respect to a given partition.
10. Program to test Riemann integrability.
11. Program to evaluate Riemann integral as a limit of sum.

Experiment 1

Program on verification of Cauchy – Riemann Equations (Cartesian Form) or test for Analyticity.

Aim: To verify of Cauchy – Riemann Equations (Cartesian Form) or to test Analyticity of given function using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
%i	Complex imaginary unit $\sqrt{-1}$
realpart (<i>expr</i>)	Returns the real part of <i>expr</i> .
imagpart (<i>expr</i>)	Returns the imaginary part of the expression <i>expr</i> .
abs (<i>z</i>)	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
cabs (<i>expr</i>)	Calculates the absolute value of an expression representing a complex number.
conjugate (<i>z</i>)	Returns the complex conjugate of <i>z</i> .
atan2 (<i>y</i> , <i>x</i>)	yields the value of $\text{atan}(y/x)$ in the interval $-\%pi$ to $\%pi$.
diff (<i>expr</i> , <i>x</i>)	Returns the first partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
radcan (<i>expr</i>)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals.
And	The logical conjunction operator.
if cond_1 then <i>expr_1</i> else <i>expr_0</i>	evaluates to <i>expr_1</i> if <i>cond_1</i> evaluates to true, otherwise the expression evaluates to <i>expr_0</i> .
print (“ <i>text</i> ”, <i>expr</i>)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
exp (<i>x</i>) or $\%e^x$	Represents the exponential function
log (<i>x</i>)	Represents the natural (base <i>e</i>) logarithm of <i>x</i> .
sin (<i>x</i>)	Trigonometric function sine of <i>x</i>
sinh (<i>x</i>)	Hyperbolic function Hyperbolic Sine of <i>x</i>
%pi;	π , an irrational number

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit)\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Analytic function: A function $f(z)$ of a complex variable $z = x + iy$ is said to be analytic at a point z_0 if it is differentiable at z_0 and at each point of some neighborhood of z_0 . $f(z)$ is analytic in a region 'R' if it is analytic at all points of 'R'. An analytic function is also called a holomorphic function or a regular function. A function which is analytic on the whole complex plane (i.e., for all $z \in \mathbb{C}$) is called an entire function.

Cauchy – Riemann (C-R) Equations in cartesian form: C-R equations give the necessary condition for a function $f(z)$ to be analytic at a point z_0 . In cartesian form, if $f(z) = u + iv$ (where, $u = u(x, y)$ and $v = v(x, y)$ are real valued functions of real variables x and y and $z = x + iy$) is analytic at z_0 then

$$u_x = v_y \text{ and } u_y = -v_x$$

hold at z_0 . These are called C-R equations. If C-R equations are not satisfied by $f(z)$ at a point, then it can't be analytic at that point.

Further, if u_x, u_y, v_x and v_y are continuous and satisfy C-R equations at a point then the function is analytic at that point.

Program:

Program to verify Cauchy – Riemann Equations in cartesian form

```
z:x+%i*y$
f(z):=given function of z$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

Worked Examples:

Problem 1. Write a program to test whether $f(z) = z + e^z$ is analytic or not, where $z = x + iy$

Program:

```

z:x+%i*y$
f(z):=z+exp(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$

if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$

f(z)= %e%i y + x + %i y + x
u= %ex cos(y) + x
v= %ex sin(y) + y
u_x= %ex cos(y) + 1
v_y= %ex cos(y) + 1
u_y= -%ex sin(y)
-v_x= -%ex sin(y)
C-R Equations are satisfied by f(z) and hence f(z) is analytic

```

Output:

```

f(z) = %e%i y + x + %i y + x
u = %ex cos(y) + x
v = %ex sin(y) + y
u_x = %ex cos(y) + 1
v_y = %ex cos(y) + 1
u_y = -%ex sin(y)
-v_x = -%ex sin(y)
C - R Equations are satisfied by f(z) and hence f(z) is analytic

```


Problem 2. Write a program to test whether $f(z) = \sin z$ is analytic or not, where $z = x + iy$

Program:

```
z:x+%i*y$
f(z):=sin(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("v_x=",v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

```
(%i16) z:x+%i*y$
f(z):=sin(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("v_x=",v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$

f(z)= sin( %i y + x )
u= sin(x) cosh(y)
v= cos(x) sinh(y)
u_x= cos(x) cosh(y)
v_y= cos(x) cosh(y)
u_y= sin(x) sinh(y)
-v_x= sin(x) sinh(y)
C-R Equations are satisfied by f(z) and hence is analytic
```

Output:

```
(z) = sin(%i y + x)
u = sin(x) cosh(y)
v = cos(x) sinh(y)
u_x = cos(x) cosh(y)
v_y = cos(x) cosh(y)
u_y = sin(x) sinh(y)
-v_x = sin(x) sinh(y)
C - R Equations are satisfied by f(z) and hence f(z) is analytic
```

Problem 3. Write a program to test whether $(z) = \log z$ is analytic or not, where $z = x + iy$

Program:

```

z:=x+%i*y$
f(z):=log(z)$
u:=realpart(f(z))$
v:=imagpart(f(z))$
u_x:=diff(u,x)$
u_y:=diff(u,y)$
v_x:=diff(v,x)$
v_y:=diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$

```

```

(%i16) z:=x+%i*y$
f(z):=log(z)$
u:=realpart(f(z))$
v:=imagpart(f(z))$
u_x:=diff(u,x)$
u_y:=diff(u,y)$
v_x:=diff(v,x)$
v_y:=diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$

f(z)= log(%i y + x)
      log(y2 + x2)
u= -----
      2
v= atan2(y, x)
      x
u_x= ----
      y2 + x2
      x
v_y= ----
      y2 + x2
      y
u_y= ----
      y2 + x2
      y
-v_x= ----
      y2 + x2
C-R Equations are satisfied by f(z) and hence is analytic

```

```

if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$

```

Output:

$$(z) = \log(iy + x)$$

$$u = \frac{\log(y^2 + x^2)}{2}$$

$$v = \text{atan2}(y, x)$$

$$u_x = \frac{x}{y^2 + x^2}$$

$$v_y = \frac{x}{y^2 + x^2}$$

$$u_y = \frac{y}{y^2 + x^2}$$

$$-v_x = \frac{y}{y^2 + x^2}$$

C – R Equations are satisfied by f(z) and hence f(z) is analytic

Problem 4. Write a program to test whether $f(z) = ze^z$ is analytic or not, where $z = x + iy$

Program:

```
z:x+%i*y$
f(z):=z*exp(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
```

```
(%i16) z:x+%i*y$
f(z):=z*exp(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$

f(z)= (%i y + x) %exy+x
u= x %ex cos(y) - %ex y sin(y)
v= x %ex sin(y) + %ex y cos(y)
u_x= -%ex y sin(y) + x %ex cos(y) + %ex cos(y)
v_y= -%ex y sin(y) + x %ex cos(y) + %ex cos(y)
u_y= -x %ex sin(y) - %ex sin(y) - %ex y cos(y)
-v_x= -x %ex sin(y) - %ex sin(y) - %ex y cos(y)
C-R Equations are satisfied by f(z) and hence is analytic
```

```
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

Output:

```
f(z) = (%iy + x)%exy+x
u = x%ex cos(y) - %exy sin(y)
v = x%ex sin(y) + %exy cos(y)
u_x = -%exy sin(y) + x%ex cos(y) + %ex cos(y)
v_y = -%exy sin(y) + x%ex cos(y) + %ex cos(y)
u_y = -x%ex sin(y) - %ex sin(y) - %exy cos(y)
-v_x = -x%ex sin(y) - %ex sin(y) - %exy cos(y)
C - R Equations are satisfied by f(z) and hence f(z) is analytic
```

Problem 5. Write a program to test whether $f(z) = \bar{z}$ is analytic or not, where $z = x + iy$ and $\bar{z}$ is the conjugate of z

Program:

```
z:x+%i*y$
f(z):=conjugate(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
```

```
(%i31) z:x+%i*y$
f(z):=conjugate(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
f(z)= x - %i y
u= x
v= -y
u_x= 1
v_y= -1
u_y= 0
-v_x= 0
C-R Equations are not satisfied by f(z) and hence f(z) is not analytic
```

```
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

Output:

```
f(z) = x - %iy
u = x
v = -y
u_x = 1
v_y = -1
u_y = 0
-v_x = 0
C - R Equations are not satisfied by f(z) and hence f(z) is not analytic
```

Problem 6. Write a program to test whether $f(z) = z \cdot \text{Im}(z)$ is analytic or not, where $z = x + iy$ and $\text{Im}(z) = \text{Imaginary part of } z$

Program:

```
z:x+%i*y$
f(z):=z*imagpart(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
```

```
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

Output:

```
f(z) = y(%iy + x)
u = xy
v = y^2
u_x = y
v_y = 2y
u_y = x
-v_x = 0
C - R Equations are not satisfied by f(z) and hence f(z) is not analytic
```

```
(%i47) z:x+%i*y$
f(z):=z*imagpart(z)$
u:realpart(f(z))$
v:imagpart(f(z))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
f(z)= y (%i y + x)
u= x y
v= y^2
u_x= y
v_y= 2 y
u_y= x
-v_x= 0
C-R Equations are not satisfied by f(z) and hence f(z) is not analytic
```

Problem 7. Write a program to test whether $(z) = \cos(x) \sin(y) + \%i \sin(x) \cos(y)$ is analytic or not,
where $z = x + iy$

Program:

```
f(x,y):=cos(x)*sin(y)+%i*sin(x)*cos(y)$
u:realpart(f(x,y))$
v:imagpart(f(x,y))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)",f(x,y))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

```
(%i15) f(x,y):=cos(x)*sin(y)+%i*sin(x)*cos(y)$
u:realpart(f(x,y))$
v:imagpart(f(x,y))$
u_x:diff(u,x)$
u_y:diff(u,y)$
v_x:diff(v,x)$
v_y:diff(v,y)$
print("f(z)",f(x,y))$
print("u=",u)$
print("v=",v)$
print("u_x=",u_x)$
print("v_y=",v_y)$
print("u_y=",u_y)$
print("-v_x=", -v_x)$
if radcan(u_x)=radcan(v_y) and radcan(u_y)=radcan(-v_x) then
print("C-R Equations are satisfied by f(z) and hence is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
f(z)= cos(x) sin(y) + %i sin(x) cos(y)
u= cos(x) sin(y)
v= sin(x) cos(y)
u_x= -sin(x) sin(y)
v_y= -sin(x) sin(y)
u_y= cos(x) cos(y)
-v_x= -cos(x) cos(y)
C-R Equations are not satisfied by f(z) and hence f(z) is not analytic
```

Output:

```
(z) = cos(x) sin(y) + %i sin(x) cos(y)
u = cos(x) sin(y)
v = sin(x) cos(y)
u_x = -sin(x) sin(y)
v_y = -sin(x) sin(y)
u_y = cos(x) cos(y)
-v_x = -cos(x) cos(y)
C - R Equations are not satisfied by f(z) and hence f(z) is not analytic
```

Exercise:

Write a program to test whether the given function (z) is analytic or not, where $z = x + iy$:

1. $(z) = \cos(z)$ (Answer: Analytic)

2. $(z) = \cosh(z)$ (Answer: Analytic)

3. $(z) = \sinh(z)$ (Answer: Analytic)

4. $f(z) = z^2$ (Answer: Analytic)

5. $f(z) = z^z$ (Answer: Analytic)

6. $f(z) = z\bar{z}$ (Answer: Not Analytic)

7. $(z) = \frac{1}{z}$ (Answer: Analytic)

8. $(z) = \frac{1}{\bar{z}}$ (Answer: Not Analytic)

9. $f(z) = z^3$ (Answer: Analytic)

10. $f(z) = \operatorname{Re}(z)$ where $\operatorname{Re}(z)$ = real part of z (Answer: Not Analytic)

11. $(z) = 3x - 4y + i(4x + 3y)$ (Answer: Analytic)

12. $(z) = iz + 4$ (Answer: Analytic)

13. $(z) = |z|$ where $|z|$ = modulus of z (Answer: Not Analytic)

[Hint: Define $f(z) = |z|$ as $f(z):=\text{cabs}(z)$ in Maxima]

Experiment 2

Program on verification of Cauchy – Riemann Equations (Polar Form) or test for Analyticity.

Aim: To verify Cauchy – Riemann Equations (Polar Form) or to test Analyticity of a given function using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
%i	Complex imaginary unit $\sqrt{-1}$
assume(r>0)	To declare that r is positive
realpart (expr)	Returns the real part of <i>expr</i> .
imagpart (expr)	Returns the imaginary part of the expression <i>expr</i> .
conjugate (z)	Returns the complex conjugate of <i>z</i> .
abs (z)	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
cabs (expr)	Calculates the absolute value of an expression representing a complex number.
atan2 (y, x)	yields the value of atan(y/x) in the interval -%pi to %pi.
diff (expr, x)	Returns the first partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
radcan (expr)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals.
and	The logical conjunction operator.
if cond_1 then expr_1 else expr_0	evaluates to <i>expr_1</i> if <i>cond_1</i> evaluates to true, otherwise the expression evaluates to <i>expr_0</i> .
print (“text”, expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
exp (x) or %e^x	Represents the exponential function
log (x)	Represents the natural (base <i>e</i>) logarithm of <i>x</i> .
sin (x)	Trigonometric function sine of <i>x</i>
sinh (x)	Hyperbolic function Hyperbolic Sine of <i>x</i>
%pi;	π , an irrational number

- Note:1. Press Shift+Enter for evaluation of commands and display of output.
2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
3. Replace dollar (\$) by semicolon (;) to see output of any input line.
4. Start each session with kill(all)\$ or quit())\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Analytic function: A function (z) of a complex variable $z = re^{i\theta} = r(\cos(\theta) + i\sin(\theta))$ is said to be analytic at a point z_0 if it is differentiable at z_0 and at each point of some neighborhood of z_0 . (z) is analytic in a region 'R' if it is analytic at all points of 'R'. An analytic function is also called a holomorphic function or a regular function. A function which is analytic on the whole complex plane (i.e., for all $z \in C$) is called an entire function.

Cauchy – Riemann (C-R) Equations in polar form: C-R equations give the necessary condition for a function (z) to be analytic at a point z_0 . In polar form, if $(z) = u + iv$ (where $u = u(r, \theta)$ and $v = v(r, \theta)$ are real valued functions of real variables r and θ and $z = re^{i\theta}$) is analytic at z_0 then

$$r \cdot u_r = v_\theta \text{ and } r \cdot v_r = -u_\theta$$

hold at z_0 . These are called C-R equations. If C-R equations are not satisfied by (z) at a point, then it can't be analytic at that point.

Further, if u_r, u_θ, v_r and v_θ are continuous and satisfy C-R equations at a point then the function is analytic at that point.

Program:

Program to verify Cauchy – Riemann Equations in polar form

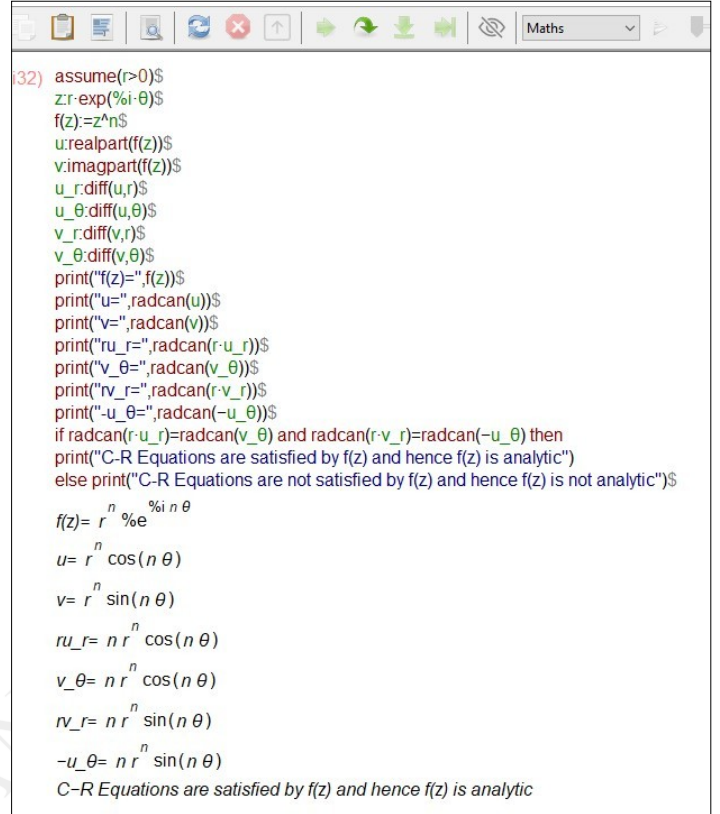
```
assume(r>0)$
z:r*exp(%i*theta)$
f(z):=given function of z$
u:realpart(f(z))$
v:imagpart(f(z))$
u_r:diff(u,r)$
u_theta:diff(u,theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
print("f(z)=",f(z))$
print("u=",radcan(u))$
print("v=",radcan(v))$
print("ru_r=",radcan(r*u_r))$
print("v_theta=",radcan(v_theta))$
print("rv_r=",radcan(r*v_r))$
print("-u_theta=",radcan(-u_theta))$
if radcan(r*u_r)=radcan(v_theta) and radcan(r*v_r)=radcan(-u_theta) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

Worked Examples:

Problem 1. Write a program to test whether $f(z) = z^n$ is analytic or not, where $z = re^{i\theta}$

Program:

```
assume(r>0)$
z:r*exp(%i*theta)$
f(z):=z^n$
u:realpart(f(z))$
v:imagpart(f(z))$
u_r:diff(u,r)$
u_theta:diff(u,theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
print("f(z)=",f(z))$
print("u=",radcan(u))$
print("v=",radcan(v))$
print("ru_r=",radcan(r*u_r))$
print("v_theta=",radcan(v_theta))$
print("rv_r=",radcan(r*v_r))$
print("-u_theta=",radcan(-u_theta))$
if radcan(r*u_r)=radcan(v_theta) and radcan(r*v_r)=radcan(-u_theta) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```



```
(32) assume(r>0)$
z:r*exp(%i*theta)$
f(z):=z^n$
u:realpart(f(z))$
v:imagpart(f(z))$
u_r:diff(u,r)$
u_theta:diff(u,theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
print("f(z)=",f(z))$
print("u=",radcan(u))$
print("v=",radcan(v))$
print("ru_r=",radcan(r*u_r))$
print("v_theta=",radcan(v_theta))$
print("rv_r=",radcan(r*v_r))$
print("-u_theta=",radcan(-u_theta))$
if radcan(r*u_r)=radcan(v_theta) and radcan(r*v_r)=radcan(-u_theta) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$

f(z)= r^n %e%i n theta
u= r^n cos(n theta)
v= r^n sin(n theta)
ru_r= n r^n cos(n theta)
v_theta= n r^n cos(n theta)
rv_r= n r^n sin(n theta)
-u_theta= n r^n sin(n theta)
C-R Equations are satisfied by f(z) and hence f(z) is analytic
```

Output:

```
f(z) = r^n %e%i n theta
u = r^n cos(n theta)
v = r^n sin(n theta)
ru_r = n r^n cos(n theta)
v_theta = n r^n cos(n theta)
rv_r = n r^n sin(n theta)
-u_theta = n r^n sin(n theta)
C - R Equations are satisfied by f(z) and hence f(z) is analytic
```

Problem 2. Write a program to test whether $f(z) = z + \frac{1}{z}$ is analytic or not, where $z = re^{i\theta}$

Program:

```
assume(r>0)$
z:r*exp(%i*theta)$
f(z):=z+1/z$
u:radcan(realpart(f(z)))$
v:radcan(imagpart(f(z)))$
u_r:diff(u,r)$
u_theta:diff(u,theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("ru_r=",radcan(r*u_r))$
print("v_theta=",radcan(v_theta))$
print("rv_r=",radcan(r*v_r))$
print("-u_theta=",radcan(-u_theta))$
if radcan(r*u_r)=radcan(v_theta) and radcan(r*v_r)=radcan(-u_theta) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

```
(%i17) assume(r>0)$
z:r*exp(%i*theta)$
f(z):=z+1/z$
u:radcan(realpart(f(z)))$
v:radcan(imagpart(f(z)))$
u_r:diff(u,r)$
u_theta:diff(u,theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("ru_r=",radcan(r*u_r))$
print("v_theta=",radcan(v_theta))$
print("rv_r=",radcan(r*v_r))$
print("-u_theta=",radcan(-u_theta))$
if radcan(r*u_r)=radcan(v_theta) and radcan(r*v_r)=radcan(-u_theta) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$

f(z) = r %e %i theta + %e - %i theta
r

u = (r^2 + 1) cos(theta)
r

v = (r^2 - 1) sin(theta)
r

ru_r = (r^2 - 1) cos(theta)
r

v_theta = (r^2 - 1) cos(theta)
r

rv_r = (r^2 + 1) sin(theta)
r

-u_theta = (r^2 + 1) sin(theta)
r

C-R Equations are satisfied by f(z) and hence f(z) is analytic
```

Output:

$$f(z) = r e^{i\theta} + \frac{e^{-i\theta}}{r}$$

$$u = \frac{(r^2 + 1) \cos(\theta)}{r}$$

$$v = \frac{(r^2 - 1) \sin(\theta)}{r}$$

$$ru_r = \frac{(r^2 - 1) \cos(\theta)}{r}$$

$$v_{\theta} = \frac{(r^2 - 1) \cos(\theta)}{r}$$

$$rv_r = \frac{(r^2 + 1) \sin(\theta)}{r}$$

$$-u_{\theta} = \frac{(r^2 + 1) \sin(\theta)}{r}$$

C - R Equations are satisfied by $f(z)$ and hence $f(z)$ is analytic

Problem 3. Write a program to test whether $(z) = \frac{1}{z}$ is analytic or not, where $z = re^{i\theta}$

Program:

```
assume(r>0)$
z:r*exp(%i*theta)$
f(z):=1/z$
u:realpart(f(z))$
v:imagpart(f(z))$
u_r:diff(u,r)$
u_theta:diff(u,theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
print("f(z)=",f(z))$
print("u=",u)$
print("v=",v)$
print("ru_r=",radcan(r*u_r))$
print("v_theta=",radcan(v_theta))$
print("rv_r=",radcan(r*v_r))$
print("-u_theta=",radcan(-u_theta))$
if radcan(r*u_r)=radcan(v_theta) and radcan(r*v_r)=radcan(-u_theta) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

$$f(z) = \frac{e^{-i\theta}}{r}$$

$$u = \frac{\cos(\theta)}{r}$$

$$v = -\frac{\sin(\theta)}{r}$$

$$ru_r = -\frac{\cos(\theta)}{r}$$

$$v_{\theta} = -\frac{\cos(\theta)}{r}$$

$$rv_r = \frac{\sin(\theta)}{r}$$

$$-u_{\theta} = \frac{\sin(\theta)}{r}$$

C-R Equations are satisfied by f(z) and hence f(z) is analytic

Output:

$$f(z) = \frac{e^{-i\theta}}{r}$$

$$u = \frac{\cos(\theta)}{r}$$

$$v = -\frac{\sin(\theta)}{r}$$

$$ru_r = -\frac{\cos(\theta)}{r}$$

$$v_{\theta} = -\frac{\cos(\theta)}{r}$$

$$rv_r = \frac{\sin(\theta)}{r}$$

$$-u_{\theta} = \frac{\sin(\theta)}{r}$$

C - R Equations are satisfied by f(z) and hence f(z) is analytic

Problem 4. Write a program to test whether $f(z) = \sqrt{r}e^{i\frac{\theta}{2}}$ is analytic or not, where $z = re^{i\theta}$

Program:

```
assume(r>0)$
f(r,θ):=r^(1/2)*exp(%i*θ/2)$
u:realpart(f(r,θ))$
v:imagpart(f(r,θ))$
u_r:diff(u,r)$
u_θ:diff(u,θ)$
v_r:diff(v,r)$
v_θ:diff(v,θ)$
print("f(z)=",f(r,θ))$
print("u=",u)$
print("v=",v)$
print("ru_r=",radcan(r*u_r))$
print("v_θ=",radcan(v_θ))$
print("rv_r=",radcan(r*v_r))$
print("-u_θ=",radcan(-u_θ))$
if radcan(r*u_r)=radcan(v_θ) and radcan(r*v_r)=radcan(-u_θ) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

```
%i16) assume(r>0)$
f(r,θ):=r^(1/2)*exp(%i*θ/2)$
u:realpart(f(r,θ))$
v:imagpart(f(r,θ))$
u_r:diff(u,r)$
u_θ:diff(u,θ)$
v_r:diff(v,r)$
v_θ:diff(v,θ)$
print("f(z)=",f(r,θ))$
print("u=",u)$
print("v=",v)$
print("ru_r=",radcan(r*u_r))$
print("v_θ=",radcan(v_θ))$
print("rv_r=",radcan(r*v_r))$
print("-u_θ=",radcan(-u_θ))$
if radcan(r*u_r)=radcan(v_θ) and radcan(r*v_r)=radcan(-u_θ) then
print("C-R Equations are satisfied by f(z) and hence f(z) is analytic")
else print("C-R Equations are not satisfied by f(z) and hence f(z) is not analytic")$
```

$$f(z) = \sqrt{r} e^{i\frac{\theta}{2}}$$

$$u = \sqrt{r} \cos\left(\frac{\theta}{2}\right)$$

$$v = \sqrt{r} \sin\left(\frac{\theta}{2}\right)$$

$$ru_r = \frac{\sqrt{r} \cos\left(\frac{\theta}{2}\right)}{2}$$

$$v_{\theta} = \frac{\sqrt{r} \cos\left(\frac{\theta}{2}\right)}{2}$$

$$rv_r = \frac{\sqrt{r} \sin\left(\frac{\theta}{2}\right)}{2}$$

$$-u_{\theta} = \frac{\sqrt{r} \sin\left(\frac{\theta}{2}\right)}{2}$$

C-R Equations are satisfied by f(z) and hence f(z) is analytic

Output:

$$f(z) = \sqrt{r} e^{i\frac{\theta}{2}}$$

$$u = \sqrt{r} \cos\left(\frac{\theta}{2}\right)$$

$$v = \sqrt{r} \sin\left(\frac{\theta}{2}\right)$$

$$ru_r = \frac{\sqrt{r} \cos\left(\frac{\theta}{2}\right)}{2}$$

$$v_{\theta} = \frac{\sqrt{r} \cos\left(\frac{\theta}{2}\right)}{2}$$

$$rv_r = \frac{\sqrt{r} \sin\left(\frac{\theta}{2}\right)}{2}$$

$$-u_{\theta} = \frac{\sqrt{r} \sin\left(\frac{\theta}{2}\right)}{2}$$

C - R Equations are satisfied by f(z) and hence f(z) is analytic

Exercise:

Write a program to test whether the given function (z) is analytic or not, where $z = re^{i\theta}$:

- | | |
|--|--------------------|
| 1. $f(z) = z^2$ | (Answer: Analytic) |
| 2. $f(z) = z^3$ | (Answer: Analytic) |
| 3. $f(z) = \sinh(z)$ | (Answer: Analytic) |
| 4. $f(z) = z^2$ | (Answer: Analytic) |
| 5. $f(z) = r^2(\cos(2\theta) + i \sin(2\theta))$ | (Answer: Analytic) |

Experiment 3

Program to check whether a function is harmonic or not.

Aim: To check whether a given function is harmonic or not using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
%i	Complex imaginary unit $\sqrt{-1}$
assume(r>0)	To declare that r is positive
realpart (expr)	Returns the real part of <i>expr</i> .
imagpart (expr)	Returns the imaginary part of the expression <i>expr</i> .
conjugate (z)	Returns the complex conjugate of <i>z</i> .
abs (z)	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
cabs (expr)	Calculates the absolute value of an expression representing a complex number.
atan2 (y, x)	yields the value of atan(<i>y/x</i>) in the interval -%pi to %pi.
diff (expr, x)	Returns the first partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
diff (expr, x, n)	Returns the n^{th} partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
radcan (expr)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals.
and	The logical conjunction operator.
if cond_1 then expr_1 else expr_0	evaluates to <i>expr_1</i> if <i>cond_1</i> evaluates to true, otherwise the expression evaluates to <i>expr_0</i> .
print (“text”, expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
exp (x) or %e^x	Represents the exponential function
log (x)	Represents the natural (base <i>e</i>) logarithm of <i>x</i> .
sin (x)	Trigonometric function sine of <i>x</i>
sinh (x)	Hyperbolic function Hyperbolic Sine of <i>x</i>
%pi;	π , an irrational number

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit())\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Harmonic function: A function ϕ is said to be harmonic if it satisfies Laplace's Equation $\nabla^2 \phi = 0$.

If ϕ is a function of x and y (Cartesian form) then ϕ is harmonic if $\phi_{xx} + \phi_{yy} = 0$. i.e.,

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0.$$

If ϕ is a function of r and θ then ϕ is harmonic if $\phi_{rr} + \frac{1}{r} \phi_r + \frac{1}{r^2} \phi_{\theta\theta} = 0$. i.e.,

$$\frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r} \frac{\partial \phi}{\partial r} + \frac{1}{r^2} \frac{\partial^2 \phi}{\partial \theta^2} = 0.$$

Harmonic property of analytic function: The real and imaginary parts of an analytic function are harmonic.

Harmonic Conjugates: Two harmonic functions u and v are said to be harmonic conjugates if $u + iv$ is analytic. Here v is called the harmonic conjugate of u and vice-versa.

Program:

Program to check whether a function in x and y (Cartesian Function) is harmonic or not.

```
u:given function of x and y$
u_xx:diff(u,x,2)$
u_yy:diff(u,y,2)$
print("u=",u)$
print("u_xx=",radcan(u_xx))$
print("u_yy=",radcan(u_yy))$
print("u_xx+u_yy=",radcan(u_xx+u_yy))$
if radcan(u_xx+u_yy)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$
```

Program to check whether a function in r and θ (Polar Function) is harmonic or not.

```
u:given function of r and theta $
u_r:diff(u,r)$
u_rr:diff(u,r,2)$
u_theta:diff(u,theta,2)$
print("u=",u)$
print("u_r=",radcan(u_r))$
print("u_rr=",radcan(u_rr))$
print("u_theta=",radcan(u_theta))$
print("u_rr+(u_r)/r+(u_theta)/r^2=",radcan(u_rr+(u_r)/r+(u_theta)/r^2))$
if radcan(u_rr+(u_r)/r+(u_theta)/r^2)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$
```


Worked Examples:

Problem 1. Write a program to check whether $u = x^2 - y^2$ is harmonic or not.

Program:

```
u:x^2-y^2$
u_xx:diff(u,x,2)$
u_yy:diff(u,y,2)$
print("u =",u)$
print("u_xx =",radcan(u_xx))$
print("u_yy =",radcan(u_yy))$
print("u_xx+u_yy =",radcan(u_xx+u_yy))$
if radcan(u_xx+u_yy)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$
```

```
(%i8) u:x^2-y^2$
u_xx:diff(u,x,2)$
u_yy:diff(u,y,2)$
print("u =",u)$
print("u_xx=",radcan(u_xx))$
print("u_yy=",radcan(u_yy))$
print("u_xx+u_yy=",radcan(u_xx+u_yy))$
if radcan(u_xx+u_yy)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$

u= x^2 - y^2
u_xx= 2
u_yy= -2
u_xx+u_yy= 0
Given function is harmonic
```

Output:

```
u = x^2 - y^2
u_xx = 2
u_yy = -2
u_xx + u_yy = 0
Given function is harmonic
```

Problem 2. Write a program to check whether $u = x^3 - 3xy^2 + 3x^2 - 3y^2 + 1$ is harmonic or not.

Program:

```
u:x^3-3*x*y^2+3*x^2-3*y^2+1$
u_xx:diff(u,x,2)$
u_yy:diff(u,y,2)$
print("u =",u)$
print("u_xx =",radcan(u_xx))$
print("u_yy =",radcan(u_yy))$
print("u_xx+u_yy =",radcan(u_xx+u_yy))$
if radcan(u_xx+u_yy)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$
```

```
(%i8) u:x^3-3*x*y^2+3*x^2-3*y^2+1$
u_xx:diff(u,x,2)$
u_yy:diff(u,y,2)$
print("u =",u)$
print("u_xx=",radcan(u_xx))$
print("u_yy=",radcan(u_yy))$
print("u_xx+u_yy=",radcan(u_xx+u_yy))$
if radcan(u_xx+u_yy)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$

u= -3 x y^2 -3 y^2 + x^3 +3 x^2 +1
u_xx= 6 x +6
u_yy= -6 x -6
u_xx+u_yy= 0
Given function is harmonic
```

Output:

$$u = -3xy^2 - 3y^2 + x^3 + 3x^2 + 1$$

$$u_{xx} = 6x + 6$$

$$u_{yy} = -6x - 6$$

$$u_{xx} + u_{yy} = 0$$

Given function is harmonic

Problem 3. Write a program to check whether $u = \log\sqrt{x+y}$ is harmonic or not.

Program:

```
u:log((x+y)^(1/2))$
u_xx:diff(u,x,2)$
u_yy:diff(u,y,2)$
print("u =",u)$
print("u_xx =",radcan(u_xx))$
print("u_yy =",radcan(u_yy))$
print("u_xx+u_yy =",radcan(u_xx+u_yy))$
if radcan(u_xx+u_yy)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$
```

```
(%i8) u:log((x+y)^(1/2))$
u_xx:diff(u,x,2)$
u_yy:diff(u,y,2)$
print("u =",u)$
print("u_xx =",radcan(u_xx))$
print("u_yy =",radcan(u_yy))$
print("u_xx+u_yy =",radcan(u_xx+u_yy))$
if radcan(u_xx+u_yy)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$

u= log(y+x)
2

u_xx= - 1
2 y^2 + 4 x y + 2 x^2

u_yy= - 1
2 y^2 + 4 x y + 2 x^2

u_xx+u_yy= - 1
y^2 + 2 x y + x^2

Given function is not harmonic
```

Output:

$$u = \frac{\log(y+x)}{2}$$

$$u_{xx} = -\frac{1}{2y^2 + 4xy + 2x^2}$$

$$u_{yy} = -\frac{1}{2y^2 + 4xy + 2x^2}$$

$$u_{xx} + u_{yy} = -\frac{1}{y^2 + 2xy + x^2}$$

Given function is not harmonic

Problem 4. Write a program to check whether $u = (r + \frac{1}{r}) \cos(\theta)$ is harmonic or not.

Program:

```
u:(r+1/r)*cos(theta)$
u_r:diff(u,r)$
u_rr:diff(u,r,2)$
```

```

u_θθ:diff(u,θ,2)$
print("u =",u)$
print("u_r =",radcan(u_r))$
print("u_rr =",radcan(u_rr))$
print("u_θθ =",radcan(u_θθ))$
print("u_rr+(u_r)/r+(u_θθ)/r^2 =",radcan(u_rr+(u_r)/r+(u_θθ)/r^2))$
if radcan(u_rr+(u_r)/r+(u_θθ)/r^2)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$

```

Output:

$$u = \left(r + \frac{1}{r}\right) \cos(\theta)$$

$$u_r = \frac{(r^2 - 1) \cos(\theta)}{r^2}$$

$$u_{rr} = \frac{2 \cos(\theta)}{r^3}$$

$$u_{\theta\theta} = -\frac{(r^2 + 1) \cos(\theta)}{r}$$

$$u_{rr} + \frac{u_r}{r} + \frac{u_{\theta\theta}}{r^2} = 0$$

Given function is harmonic

```

(%i10) u:(r+1/r)*cos(theta)$
u_r:diff(u,r)$
u_rr:diff(u,r,2)$
u_θθ:diff(u,θ,2)$
print("u =",u)$
print("u_r =",radcan(u_r))$
print("u_rr =",radcan(u_rr))$
print("u_θθ =",radcan(u_θθ))$
print("u_rr+(u_r)/r+(u_θθ)/r^2 =",radcan(u_rr+(u_r)/r+(u_θθ)/r^2))$
if radcan(u_rr+(u_r)/r+(u_θθ)/r^2)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$

u = (r + 1/r) cos(θ)
u_r = ((r^2 - 1) cos(θ)) / r^2
u_rr = (2 cos(θ)) / r^3
u_θθ = -((r^2 + 1) cos(θ)) / r
u_rr + (u_r)/r + (u_θθ)/r^2 = 0
Given function is harmonic

```

Problem 5. Write a program to check whether $v = -\frac{\sin(\theta)}{r}$ is harmonic or not.

Program:

```

v:-sin(θ)/r$
v_r:diff(v,r)$
v_rr:diff(v,r,2)$
v_θθ:diff(v,θ,2)$
print("v =",v)$
print("v_r =",radcan(v_r))$
print("v_rr =",radcan(v_rr))$
print("v_θθ =",radcan(v_θθ))$
print("v_rr+(v_r)/r+(v_θθ)/r^2 =",radcan(v_rr+(v_r)/r+(v_θθ)/r^2))$
if radcan(v_rr+(v_r)/r+(v_θθ)/r^2)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$

```

Output:

$$v = -\frac{\sin(\theta)}{r}$$

$$v_r = \frac{\sin(\theta)}{r^2}$$

$$v_{rr} = -\frac{2 \sin(\theta)}{r^3}$$

$$v_{\theta\theta} = \frac{\sin(\theta)}{r}$$

$$v_{rr} + \frac{v_r}{r} + \frac{v_{\theta\theta}}{r^2} = 0$$

Given function is harmonic

```
(%i10) v:=-sin(theta)/r$
v_r:=diff(v,r)$
v_rr:=diff(v_r,r)$
v_theta:=diff(v,theta)$
print("v =",v)$
print("v_r =",radcan(v_r))$
print("v_rr =",radcan(v_rr))$
print("v_theta =",radcan(v_theta))$
print("v_rr+(v_r)/r+(v_theta)/r^2 =",radcan(v_rr+(v_r)/r+(v_theta)/r^2))$
if radcan(v_rr+(v_r)/r+(v_theta)/r^2)=0 then
print("Given function is harmonic")
else print("Given function is not harmonic")$

v = - sin(theta)/r
v_r = sin(theta)/r^2
v_rr = - 2 sin(theta)/r^3
v_theta = sin(theta)/r
v_rr+(v_r)/r+(v_theta)/r^2 = 0
Given function is harmonic
```

Exercise:

Write a program to check whether the given functions are harmonic or not:

1. $u = \log \sqrt{x^2 + y^2}$

(Answer: Harmonic)

2. $v = \cos(x) \sinh(y)$

(Answer: Harmonic)

3. $u = x^2 + 4x - y^2 + 2y$

(Answer: Harmonic)

4. $u = e(x \cos(y) - y \sin(y))$

(Answer: Harmonic)

5. $u = 3x^2 + 2xy - 2y^2$

(Answer: Not Harmonic)

6. $v = -r^3 \sin(3\theta)$

(Answer: Harmonic)

7. $u = r^2 \cos(3\theta)$

(Answer: Not Harmonic)

8. $u = \frac{\cos(\theta)}{r}$

(Answer: Harmonic)

Experiment 4

Program to construct analytic functions (Milne-Thomson Method).

Aim: To construct analytic function from its real/imaginary part by applying Milne-Thomson Method using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
%i	Complex imaginary unit $\sqrt{-1}$
assume(r>0)	To declare that r is positive
realpart (expr)	Returns the real part of <i>expr</i> .
imagpart (expr)	Returns the imaginary part of the expression <i>expr</i> .
conjugate (z)	Returns the complex conjugate of <i>z</i> .
abs (z)	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
cabs (expr)	Calculates the absolute value of an expression representing a complex number.
atan2 (y, x)	yields the value of $\text{atan}(y/x)$ in the interval $-\pi$ to π .
diff (expr, x)	Returns the first partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
subst (a, b, c)	Substitutes a for b in c
subst ([eq_1, ..., eq_k], expr)	For each equation, the right side will be substituted for the left in the expression <i>expr</i> .
radcan (expr)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals.
if cond_1 then expr_1 else expr_0	evaluates to <i>expr_1</i> if <i>cond_1</i> evaluates to true, otherwise the expression evaluates to <i>expr_0</i> .
print (“text”, expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
exp (x) or %e^x	Represents the exponential function
log (x)	Represents the natural (base <i>e</i>) logarithm of <i>x</i> .
sin (x)	Trigonometric function sine of <i>x</i>
sinh (x)	Hyperbolic function Hyperbolic Sine of <i>x</i>
%pi;	π , an irrational number

- Note: 1. Press Shift+Enter for evaluation of commands and display of output.
2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
3. Replace dollar (\$) by semicolon (;) to see output of any input line.
4. Start each session with kill(all)\$ or quit)\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Milne-Thomson Method of constructing analytic function from its real or imaginary part:

Cartesian function:

Step 1. If the real part u is given then find $f'(z) = u_x - iu_y$. If the imaginary part v is given then find

$$f'(z) = v_y + iv_x.$$

Step 2. Substitute $x = z, y = 0$ in $f'(z)$ to express it in z .

Step 3. Integrate $f'(z)$ with respect to z to get $f(z)$.

Polar function:

Step 1. If the real part u is given then find $f'(z) = e^{-i\theta} (u_r - i \frac{u_\theta}{r})$. If the imaginary part v is given

$$\text{then find } f'(z) = e^{-i\theta} (\frac{v_\theta}{r} + iv_r).$$

Step 2. Substitute $r = z, \theta = 0$ in $f'(z)$ to express it in z .

Step 3. Integrate $f'(z)$ with respect to z to get $f(z)$.

Program:

Program to construct analytic function when real part is given in x and y (Cartesian Function).

```
u:given function of x and y$
u_x:diff(u,x)$
u_y:diff(u,y)$
f:radcan(subst([x=z, y=0], u_x-%i*u_y))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Program to construct analytic function when imaginary part is given in x and y (Cartesian Function).

```
v:given function of x and y$
v_x:diff(v,x)$
v_y:diff(v,y)$
f:radcan(subst([x=z, y=0], v_y+%i*v_x))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Program to construct analytic function when real part is given in r and θ (Polar Function).

```
u: given function in r and  $\theta$  $
u_r:diff(u,r)$
u_theta:diff(u, $\theta$ )$
f:radcan(subst([r=z,  $\theta$ =0], exp(-%i* $\theta$ )*(u_r-%i*u_theta/r)))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Program to construct analytic function when imaginary part is given in r and θ (Polar Function).

```
v:given function in r and  $\theta$  $
v_r:diff(v,r)$
v_theta:diff(v, $\theta$ )$
f:radcan(subst([r=z,  $\theta$ =0], exp(-%i* $\theta$ )*(v_theta/r+%i*v_r)))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Worked Examples:

Problem 1. Write a program to construct an analytic function whose real part is $u = \log \sqrt{x^2 + y^2}$.

Program:

```
u:1/2*log(x^2+y^2)$
u_x:diff(u,x)$
u_y:diff(u,y)$
f:radcan(subst([x=z, y=0], u_x-%i*u_y))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

```
(%i8) u:1/2*log(x^2+y^2)$
u_x:diff(u,x)$
u_y:diff(u,y)$
f:radcan(subst([x=z, y=0], u_x-%i*u_y))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$

      log(y2+x2)
      -----
             2

f(z)= 1
      z

Required Analytic function is f(z)= log(z)
```

Output:

$$u = \frac{\log(y^2 + x^2)}{2}$$

$$f'(z) = \frac{1}{z}$$

Required Analytic function is $f(z) = \log(z)$

Problem 2. Write a program to construct an analytic function whose real part is $u = \sin(x) \cosh(y)$.

Program:

```
u:sin(x)*cosh(y)$
u_x:diff(u,x)$
u_y:diff(u,y)$
f:radcan(subst([x=z, y=0], u_x-%i*u_y))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

```
) u:sin(x)*cosh(y)$
u_x:diff(u,x)$
u_y:diff(u,y)$
f:radcan(subst([x=z, y=0], u_x-%i*u_y))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
u= sin(x) cosh(y)
f(z)= cos(z)
Required Analytic function is f(z)= sin(z)
```

Output:

$u = \sin(x) \cosh(y)$
 $f'(z) = \cos(z)$
 Required Analytic function is $f(z) = \sin(z)$

Problem 3. Write a program to construct an analytic function whose real part is $u = \frac{\sin(2x)}{\cosh(2y) - \cos(2x)}$

Program:

```
u:sin(2*x)/(cosh(2*y)-cos(2*x))$
u_x:diff(u,x)$
u_y:diff(u,y)$
f:trigreduce(trigrat(subst([x=z, y=0], u_x-%i*u_y))$
F:trigreduce(trigrat(integrate(f,z)))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

```
(%i8) u:sin(2*x)/(cosh(2*y)-cos(2*x))$
u_x:diff(u,x)$
u_y:diff(u,y)$
f:trigreduce(trigrat(subst([x=z, y=0], u_x-%i*u_y))$
F:trigreduce(trigrat(integrate(f,z)))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
u= sin(2 x)
cosh(2 y)-cos(2 x)
f(z)= 2
cos(2 z)-1
Required Analytic function is f(z)= cot(z)
```

Output:

$u = \frac{\sin(2x)}{\cosh(2y) - \cos(2x)}$
 $f'(z) = \frac{2}{\cos(2z) - 1}$
 Required Analytic function is $f(z) = \cot(z)$

Problem 4. Construct an analytic function whose imaginary part is $v = e(x \sin(y) + y \cos(y))$

Program:

```
v:exp(x)*(x*sin(y)+y*cos(y))$
v_x:diff(v,x)$
v_y:diff(v,y)$
f:radcan(subst([x=z, y=0], v_y+%i*v_x))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

```
(%i8) v:exp(x)*(x*sin(y)+y*cos(y))$
v_x:diff(v,x)$
v_y:diff(v,y)$
f:radcan(subst([x=z, y=0], v_y+%i*v_x))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$

v= %ex (x sin(y)+y cos(y))
f(z)= (z+1) %ez
Required Analytic function is f(z)= z %ez
```

Output:

$v = e(x \sin(y) + y \cos(y))$
 $f'(z) = (z + 1)e^z$
 Required Analytic function is $f(z) = ze^z$

Problem 5. Construct an analytic function whose imaginary part is $v = -\frac{\sinh(2y)}{\cosh(2y) - \cos(2x)}$

Program:

```
v:-sinh(2*y)/(cosh(2*y)-cos(2*x))$
v_x:diff(v,x)$
v_y:diff(v,y)$
f:radcan(subst([x=z, y=0], v_y+%i*v_x))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
print(" That is, f(z)=",trigreduce(trigrat(F)))$
```

```
(%i9) v:-sinh(2*y)/(cosh(2*y)-cos(2*x))$
v_x:diff(v,x)$
v_y:diff(v,y)$
f:radcan(subst([x=z, y=0], v_y+%i*v_x))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
print(" That is, f(z)=",trigreduce(trigrat(F)))$

v= -  $\frac{\sinh(2y)}{\cosh(2y) - \cos(2x)}$ 
f(z)=  $\frac{2}{\cos(2z) - 1}$ 
Required Analytic function is f(z)=  $\frac{\cos(2z) + 1}{\sin(2z)}$ 
That is, f(z)= cot(z)
```

Output:

$v = -\frac{\sinh(2y)}{\cosh(2y) - \cos(2x)}$
 $f'(z) = \frac{2}{\cos(2z) - 1}$
 Required Analytic function is $f(z) = \frac{\cos(2z) + 1}{\sin(2z)}$
 That is, $f(z) = \cot(z)$

Problem 6. Write a program to construct an analytic function whose real part is $u = \frac{\cos(\theta)}{r}$

Program:

```
u:cos(θ)/r$
u_r:diff(u,r)$
u_θ:diff(u,θ)$
f:radcan(subst([r=z, θ=0], exp(-%i*θ)*(u_r-%i*u_θ/r)))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Output:

$$u = \frac{\cos(\theta)}{r}$$

$$f'(z) = -\frac{1}{z^2}$$

$$\text{Required Analytic function is } f(z) = \frac{1}{z}$$

```
u:cos(θ)/r$
u_r:diff(u,r)$
u_θ:diff(u,θ)$
f:radcan(subst([r=z, θ=0], exp(-%i*θ)*(u_r-%i*u_θ/r)))$
F:radcan(integrate(f,z))$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$

u= cos(θ)
r

f'(z)= - 1
z2

Required Analytic function is f(z)= 1
z
```

Problem 7. Write a program to construct an analytic function whose real part is $u = (r + \frac{1}{r}) \cos(\theta)$

Program:

```
u:(r+1/r)*cos(θ)$
u_r:diff(u,r)$
u_θ:diff(u,θ)$
f:subst([r=z, θ=0], exp(-%i*θ)*(u_r-%i*u_θ/r))$
F:integrate(f,z)$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Output:

$$u = (r + \frac{1}{r}) \cos(\theta)$$

$$f'(z) = 1 - \frac{1}{z^2}$$

$$\text{Required Analytic function is } f(z) = z + \frac{1}{z}$$

```
u:(r+1/r)*cos(θ)$
u_r:diff(u,r)$
u_θ:diff(u,θ)$
f:subst([r=z, θ=0], exp(-%i*θ)*(u_r-%i*u_θ/r))$
F:integrate(f,z)$
print("u=",u)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$

u= (r+1/r)cos(θ)

f'(z)= 1 - 1
z2

Required Analytic function is f(z)= z+1
z
```

Problem 8. Write a program to construct an analytic function whose imaginary part is $v = \left(r - \frac{1}{r}\right) \sin(\theta)$

Program:

```
v:(r-1/r)*sin(theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
f:radcan(subst([r=z, theta=0], exp(-%i*theta)*(v_theta/r+%i*v_r)))$
F:integrate(f,z)$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Output:

$$v = \left(r - \frac{1}{r}\right) \sin(\theta)$$

$$f'(z) = \frac{z^2 - 1}{z^2}$$

Required Analytic function is $f(z) = z + \frac{1}{z}$

```
v:(r-1/r)*sin(theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
f:radcan(subst([r=z, theta=0], exp(-%i*theta)*(v_theta/r+%i*v_r)))$
F:integrate(f,z)$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$

v= (r - 1/r) sin(theta)

f'(z)= (z^2 - 1)/z^2

Required Analytic function is f(z)= z + 1/z
```

Problem 9. Write a program to construct an analytic function whose imaginary part is $v = r^2 \cos(2\theta)$

Program:

```
v:r^2*cos(2*theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
f:radcan(subst([r=z, theta=0], exp(-%i*theta)*(v_theta/r+%i*v_r)))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$
```

Output:

$$v = r^2 \cos(2\theta)$$

$$f'(z) = 2\%iz$$

Required Analytic function is $f(z) = \%iz^2$

```
v:r^2*cos(2*theta)$
v_r:diff(v,r)$
v_theta:diff(v,theta)$
f:radcan(subst([r=z, theta=0], exp(-%i*theta)*(v_theta/r+%i*v_r)))$
F:radcan(integrate(f,z))$
print("v=",v)$
print("f(z)=",f)$
print(" Required Analytic function is f(z)=",F)$

v= r^2 cos(2 theta)

f'(z)= 2 %i z

Required Analytic function is f(z)= %i z^2
```

Exercise:

Write a program to construct an analytic function from the given real part (u) or imaginary part (v):

1. $u = x^2 - y^2$

(Answer: $(z) = z^2$)

2. $u = \frac{x^4 - y^4 - 2x}{x^2 + y^2}$

(Answer: $(z) = \frac{z^3 - 2}{z}$)

3. $u = e^2(x \cos(2y) - y \sin(2y))$

(Answer: $f(z) = ze^{2z}$)

4. $u = r^2 \cos(2\theta)$

(Answer: $f(z) = z^2$)

5. $v = \cos(x) \cosh(y)$

(Answer: $f(z) = \%i \cos(z)$)

6. $v = \frac{x-y}{y^2+x^2}$

(Answer: $(z) = \frac{\%i+1}{z}$)

7. $v = -\frac{\sin(\theta)}{r}$

(Answer: $(z) = \frac{1}{z}$)

8. $v = \sqrt{r} \sin\left(\frac{\theta}{2}\right)$

(Answer: $(z) = \sqrt{z}$)

Experiment 5

Program to find cross-ratio of points and related concepts.

Aim: To find the cross ratio of given points and bilinear transformation related problems using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
%i	Complex imaginary unit $\sqrt{-1}$
assume(r>0)	To declare that r is positive
realpart (expr)	Returns the real part of <i>expr</i> .
imagpart (expr)	Returns the imaginary part of the expression <i>expr</i> .
conjugate (z)	Returns the complex conjugate of <i>z</i> .
abs (z)	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
cabs (expr)	Calculates the absolute value of an expression representing a complex number.
atan2 (y, x)	yields the value of atan(y/x) in the interval $-\%pi$ to $\%pi$.
rectform (expr)	Returns an expression $a + b \%i$ equivalent to <i>expr</i> , such that <i>a</i> and <i>b</i> are purely real
:=	The function definition operator.
$f(x_1, \dots, x_n) := expr$	Defines a function named <i>f</i> with arguments $x_1, \dots, x_n$ and function body <i>expr</i>
subst ([eq_1, ..., eq_k], expr)	For each equation, the right side will be substituted for the left in the expression <i>expr</i> .
radcan (expr)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals.
if cond_1 then expr_1 else expr_0	evaluates to <i>expr_1</i> if <i>cond_1</i> evaluates to true, otherwise the expression evaluates to <i>expr_0</i> .
print (“text”, expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
solve (expr, x)	Solves the algebraic equation <i>expr</i> for the variable <i>x</i> and returns a list of solution equations in <i>x</i>
inf	inf represents real positive infinity
%pi;	π , an irrational number

- Note:**
1. Press Shift+Enter for evaluation of commands and display of output.
 2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
 3. Replace dollar (\$) by semicolon (;) to see output of any input line.
 4. Start each session with kill(all)\$ or quit(\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Cross Ratio:

Let z_1, z_2, z_3, z_4 be four distinct points in the extended complex plane $\mathbb{C}_\infty$ then the following

$$\frac{(z_1 - z_2)(z_3 - z_4)}{(z_2 - z_3)(z_4 - z_1)}$$

is called the cross ratio of z_1, z_2, z_3, z_4 and is denoted by (z_1, z_2, z_3, z_4) . By performing permutations on the four points z_1, z_2, z_3, z_4 we get $4! = 24$ cross ratios but only six of them are distinct. Thus, distinct cross ratios of z_1, z_2, z_3, z_4 are given by:

$$\begin{aligned} 1) & \frac{(z_1 - z_2)(z_3 - z_4)}{(z_2 - z_3)(z_4 - z_1)} & 2) & \frac{(z_1 - z_2)(z_4 - z_3)}{(z_2 - z_4)(z_3 - z_1)} & 3) & \frac{(z_1 - z_3)(z_2 - z_4)}{(z_3 - z_2)(z_4 - z_1)} \\ 4) & \frac{(z_1 - z_3)(z_4 - z_2)}{(z_3 - z_4)(z_2 - z_1)} & 5) & \frac{(z_1 - z_4)(z_2 - z_3)}{(z_4 - z_2)(z_3 - z_1)} & 6) & \frac{(z_1 - z_4)(z_3 - z_2)}{(z_4 - z_3)(z_2 - z_1)} \end{aligned}$$

The values of these distinct cross ratios are related. If the first one is λ then the others will be:

$$\frac{\lambda}{1-\lambda}, 1-\lambda, \frac{1-\lambda}{\lambda}, \frac{1}{1-\lambda}, \frac{1}{\lambda}$$

of z_1, z_2, z_3, z_4 as:

$$(z_1, z_2, z_3, z_4) = \frac{(z_1 - z_2)(z_3 - z_4)}{(z_2 - z_3)(z_4 - z_1)}$$

Further note that,

$$(z_1, z_2, z_3, z_4) = \frac{(z_1 - z_2)}{(z_3 - z_1)} \text{ if } z_4 = \infty$$

$$(z_1, z_2, z_3, z_4) = \frac{(z_1 - z_2)}{(z_1 - z_4)} \text{ if } z_3 = \infty$$

$$(z_1, z_2, z_3, z_4) = \frac{(z_3 - z_4)}{(z_1 - z_4)} \text{ if } z_2 = \infty$$

$$(z_1, z_2, z_3, z_4) = \frac{(z_3 - z_4)}{(z_3 - z_2)} \text{ if } z_1 = \infty$$

The linear transformation: A transformation of the form $w = az + b$, is called a linear transformation, where a and b are complex constants.

Bilinear transformation: A transformation of the form $w = \frac{az+b}{cz+d}$ is called a Bilinear transformation or linear fractional transformation, where a, b, c, d are complex constants and $ad - bc \neq 0$. This transformation is linear in both w and z and hence it is bilinear transformation. It is also called Mobius transformation. Inverse transformation is $z = \frac{-dw+b}{cw-a}$

Preservation of cross ratio: Any bilinear transformation preserves cross ratio. i.e., cross ratio is invariant under a bilinear transformation. This fact can be used to find a bilinear transformation which maps z_1, z_2, z_3 to w_1, w_2, w_3 respectively by using:

$$\frac{(z - z_1)(z_2 - z_3)}{(z_1 - z_2)(z - z_3)} = \frac{(w - w_1)(w_2 - w_3)}{(w_1 - w_2)(w - w_3)}$$

i.e.,

$$(w - w_1)(w_2 - w_3)(z_1 - z_2)(z_3 - z) = (w_1 - w_2)(w_3 - w)(z - z_1)(z_2 - z_3)$$

Further note that,

$$(w - w_1)(w_2 - w_3)(z_1 - z_2) = -(w_1 - w_2)(w_3 - w)(z - z_1) \text{ if } z_3 = \infty$$

$$(w - w_1)(w_2 - w_3)(z_3 - z) = -(w_1 - w_2)(w_3 - w)(z - z_1) \text{ if } z_2 = \infty$$

$$(w - w_1)(w_2 - w_3)(z_3 - z) = -(w_1 - w_2)(w_3 - w)(z_2 - z_3) \text{ if } z_1 = \infty$$

$$(w - w_1)(z_1 - z_2)(z_3 - z) = -(w_1 - w_2)(z - z_1)(z_2 - z_3) \text{ if } w_3 = \infty$$

$$(w - w_1)(z_1 - z_2)(z_3 - z) = -(w_3 - w)(z - z_1)(z_2 - z_3) \text{ if } w_2 = \infty$$

$$(w_2 - w_3)(z_1 - z_2)(z_3 - z) = -(w_3 - w)(z - z_1)(z_2 - z_3) \text{ if } w_1 = \infty$$

Program:

Program to find the cross ratio of four points (z_1, z_2, z_3, z_4).

```
z1:given value of z1$
z2: given value of z2$
z3: given value of z3$
z4: given value of z4$
CR:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
print("Cross Ratio (z1, z2, z3, z4) =", CR)$
```

Program to check invariance of cross ratio of four points (z_1, z_2, z_3, z_4) in a bilinear transformation.

```
w(z):= given transformation$
z1:given value of z1$
z2: given value of z2$
z3: given value of z3$
z4: given value of z4$
CR1:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
CR2:radcan(((w(z1)-w(z2))*(w(z3)-w(z4)))/((w(z2)-w(z3))*(w(z4)-w(z1))))$
print("Given transformation is w(z) =", w(z))$
print("Cross Ratio (z1, z2, z3, z4) =", rectform(CR1))$
print("Cross Ratio (w1, w2, w3, w4) =", rectform(CR2))$
if rectform(CR1)=rectform(CR2) then
print("Cross ratio is invariant in the given transformation")
else print("Cross ratio is not preserved in the given transformation")$
```

Program to find bilinear transformation which maps z_1, z_2, z_3 to w_1, w_2, w_3 respectively

```

z1: given value of z1$
z2: given value of z2$
z3: given value of z3$
w1: given value of w1$
w2: given value of w2$
w3: given value of w3$
eq:(w-w1)*(w2-w3)*(z1-z2)*(z3-z)=(w1-w2)*(w3-w)*(z-z1)*(z2-z3)$
print("Required Bilinear Transformation is w(z)=", radcan(rhs(solve(eq,w)[1])))$

```

Note: When one of the given points is infinity, use correct expressions given in definitions section.

Worked Examples:

Problem 1. Write a program to find the cross ratio of four points $i, 1, 2 - i, 3$.

Program:

```

z1:%i$
z2:1$
z3:2-%i$
z4:3$
CR:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
print("Cross Ratio (z1, z2, z3, z4) =", CR)$

```

```

z1:%i$
z2:1$
z3:2-%i$
z4:3$
CR:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
print("Cross Ratio (z1, z2, z3, z4) =", CR)$

```

$$\text{Cross Ratio (z1, z2, z3, z4)} = \frac{\%i + 1}{\%i - 3}$$

Output:

$$\text{Cross Ratio (z1, z2, z3, z4)} = \frac{\%i + 1}{\%i - 3}$$

Problem 2. Write a program to find the cross ratio of four points $i, 1, 2, \infty$.

Program:

```

z1:%i$
z2:1$
z3:2$
z4:inf$
CR:radcan((z1-z2)/(z3-z2))$
print("Cross Ratio (z1, z2, z3, z4) =", CR)$

```

```

z1:%i$
z2:1$
z3:2$
z4:inf$
CR:radcan((z1-z2)/(z3-z2))$
print("Cross Ratio (z1, z2, z3, z4) =", CR)$

```

$$\text{Cross Ratio (z1, z2, z3, z4)} = \%i - 1$$

Output:

$$\text{Cross Ratio (z1, z2, z3, z4)} = \%i - 1$$

Problem 3. Write a program to check invariance of cross ratio of four points (i , 1 , $2 - i$, 3) in the

$$\text{transformation } w = \frac{1+iz}{1-iz}$$

Program:

```
w(z):=(1+%i*z)/(1-%i*z)$
z1:%i$
z2:1$
z3:2-%i$
z4:3$
CR1:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
CR2:radcan(((w(z1)-w(z2))*(w(z3)-w(z4)))/((w(z2)-w(z3))*(w(z4)-w(z1))))$
print("Given transformation is w(z) =", w(z))$
print("Cross Ratio (z1, z2, z3, z4) =", rectform(CR1))$
print("Cross Ratio (w1, w2, w3, w4) =", rectform(CR2))$
if rectform(CR1)=rectform(CR2) then
print("Cross ratio is invariant in the given transformation")
else print("Cross ratio is not preserved in the given transformation")$
```

Output:

$$\text{Given transformation is } (z) = \frac{\%iz + 1}{1 - \%iz}$$

$$\text{Cross Ratio } (z1, z2, z3, z4) = -\frac{2\%i}{5} - \frac{1}{5}$$

$$\text{Cross Ratio } (w1, w2, w3, w4) = -\frac{2\%i}{5} - \frac{1}{5}$$

Cross ratio is invariant in the given transformation

```
%i11) w(z):=(1+%i*z)/(1-%i*z)$
z1:%i$
z2:1$
z3:2-%i$
z4:3$
CR1:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
CR2:radcan(((w(z1)-w(z2))*(w(z3)-w(z4)))/((w(z2)-w(z3))*(w(z4)-w(z1))))$
print("Given transformation is w(z) =", w(z))$
print("Cross Ratio (z1, z2, z3, z4) =", rectform(CR1))$
print("Cross Ratio (w1, w2, w3, w4) =", rectform(CR2))$
if rectform(CR1)=rectform(CR2) then
print("Cross ratio is invariant in the given transformation")
else print("Cross ratio is not preserved in the given transformation")$

Given transformation is w(z) = \frac{\%i z + 1}{1 - \%i z}

Cross Ratio (z1, z2, z3, z4) = -\frac{2 \%i}{5} - \frac{1}{5}

Cross Ratio (w1, w2, w3, w4) = -\frac{2 \%i}{5} - \frac{1}{5}

Cross ratio is invariant in the given transformation
```

Problem 4. Write a program to check invariance of cross ratio of four points $(2, i, -2, -i)$ in the transformation $w = \frac{5-4z}{4z-2}$

Program:

```
w(z):=(5-4*z)/(4*z-2)$
z1:2$
z2:%i$
z3:-2$
z4:%i$
CR1:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
CR2:radcan(((w(z1)-w(z2))*(w(z3)-w(z4)))/((w(z2)-w(z3))*(w(z4)-w(z1))))$
print("Given transformation is w(z) =", w(z))$
print("Cross Ratio (z1, z2, z3, z4) =", rectform(CR1))$
print("Cross Ratio (w1, w2, w3, w4) =", rectform(CR2))$
if rectform(CR1)=rectform(CR2) then
print("Cross ratio is invariant in the given transformation")
else print("Cross ratio is not preserved in the given transformation")$
```

Output:

Given transformation is $w(z) = \frac{5-4z}{4z-2}$

Cross Ratio $(z1, z2, z3, z4) = -\frac{24\%i}{25} - \frac{7}{25}$

Cross Ratio $(w1, w2, w3, w4) = -\frac{24\%i}{25} - \frac{7}{25}$

Cross ratio is invariant in the given transformation

```
1) w(z):=(5-4*z)/(4*z-2)$
z1:2$
z2:%i$
z3:-2$
z4:%i$
CR1:radcan(((z1-z2)*(z3-z4))/((z2-z3)*(z4-z1)))$
CR2:radcan(((w(z1)-w(z2))*(w(z3)-w(z4)))/((w(z2)-w(z3))*(w(z4)-w(z1))))$
print("Given transformation is w(z) =", w(z))$
print("Cross Ratio (z1, z2, z3, z4) =", rectform(CR1))$
print("Cross Ratio (w1, w2, w3, w4) =", rectform(CR2))$
if rectform(CR1)=rectform(CR2) then
print("Cross ratio is invariant in the given transformation")
else print("Cross ratio is not preserved in the given transformation")$

Given transformation is w(z) =  $\frac{5-4z}{4z-2}$ 

Cross Ratio (z1, z2, z3, z4) =  $-\frac{24\%i}{25} - \frac{7}{25}$ 

Cross Ratio (w1, w2, w3, w4) =  $-\frac{24\%i}{25} - \frac{7}{25}$ 

Cross ratio is invariant in the given transformation
```

Problem 5. Write a program to find the bilinear transformation which maps $-i, 0, i$ to $-1, i, 1$ respectively.

Program:

```

z1:-%i$
z2:0$
z3:%i$
w1:-1$
w2:%i$
w3:1$
eq:(w-w1)*(w2-w3)*(z1-z2)*(z3-z)=(w1-w2)*(w3-w)*(z-z1)*(z2-z3)$
print("Required Bilinear Transformation is w(z)=",factor(rhs(solve(eq,w)[1])))$

Required Bilinear Transformation is w(z)= -  $\frac{\%i(z-1)}{z+1}$ 

eq:(w-w1)*(w2-w3)*(z1-z2)*(z3-z)=(w1-w2)*(w3-w)*(z-z1)*(z2-z3)$
print("Required Bilinear Transformation is w(z)=",factor(rhs(solve(eq,w)[1])))$

```

Output:

$$\text{Required Bilinear Transformation is } w = - \frac{\%i(z-1)}{z+1}$$

Problem 6. Write a program to find the bilinear transformation which maps $0, 1, \infty$ to $-5, -1, 3$ respectively.

Program:

```

z1:0$
z2:1$
z3:inf$
w1:-5$
w2:-1$
w3:3$
eq:(w-w1)*(w2-w3)*(z1-z2)=- (w1-w2)*(w3-w)*(z-z1)$
print("Required Bilinear Transformation is w(z)=",radcan(rhs(solve(eq,w)[1])))$

Required Bilinear Transformation is w(z)=  $\frac{3z-5}{z+1}$ 

eq:(w-w1)*(w2-w3)*(z1-z2)=- (w1-w2)*(w3-w)*(z-z1)$
print("Required Bilinear Transformation is w(z)=",radcan(rhs(solve(eq,w)[1])))$

```

Output:

$$\text{Required Bilinear Transformation is } w = \frac{3z-5}{z+1}$$

Problem 7. Write a program to find the bilinear transformation which maps $1, i, -1$ to $0, 1, \infty$ respectively.

Program:

```

z1:1$
z2:%i$
z3:-1$
w1:0$
w2:1$
w3:inf$
eq:(w-w1)*(z1-z2)*(z3-z)=- (w1-w2)*(z-z1)*(z2-z3)$
print("Required Bilinear Transformation is w(z)=",factor(rhs(solve(eq,w)[1])))$
print("That is, w(z)=",rectform(factor(rhs(solve(eq,w)[1]))))$

Required Bilinear Transformation is w(z)=  $\frac{(\%i+1)(z-1)}{(\%i-1)(z+1)}$ 

That is, w(z)=  $-\frac{\%i(z-1)}{z+1}$ 

eq:(w-w1)*(z1-z2)*(z3-z)=- (w1-w2)*(z-z1)*(z2-z3)$
print("Required Bilinear Transformation is w(z)=",factor(rhs(solve(eq,w)[1])))$
print("That is, w(z)=",rectform(factor(rhs(solve(eq,w)[1]))))$

```

Output:

$$\text{Required Bilinear Transformation is } w = \frac{(\%i+1)(z-1)}{(\%i-1)(z+1)}$$

$$\text{That is, } w(z) = -\frac{\%i(z-1)}{z+1}$$

Problem 8. Write a program to find the bilinear transformation which maps $\infty, i, 0$ to $0, i, \infty$ respectively.

Program:

```

z1:inf$
z2:%i$
z3:0$
w1:0$
w2:%i$
w3:inf$
eq:(w-w1)*(z3-z)=(w1-w2)*(z2-z3)$
print("Required Bilinear Transformation is w(z)=",radcan(rhs(solve(eq,w)[1])))$

Required Bilinear Transformation is w(z)=  $-\frac{1}{z}$ 

eq:(w-w1)*(z3-z)=(w1-w2)*(z2-z3)$
print("Required Bilinear Transformation is w(z)=",radcan(rhs(solve(eq,w)[1])))$

```

Output:

$$\text{Required Bilinear Transformation is } w = -\frac{1}{z}$$

Exercise:

Write a program to find the cross ratio of four points (z_1, z_2, z_3, z_4).

1. $z_1 = 0, z_2 = 3, z_3 = 4, z_4 = 6$ (Answer: *Cross Ratio* = -1)
2. $z_1 = 1, z_2 = i, z_3 = 0, z_4 = -i$ (Answer: *Cross Ratio* = $\frac{i-1}{i+1}$)
3. $z_1 = 0, z_2 = i, z_3 = \infty, z_4 = -1$ (Answer: *Cross Ratio* = $-i$)

Write a program to check invariance of cross ratio of four points (z_1, z_2, z_3, z_4)

in the given transformation w .

1. $z_1 = 5, z_2 = 3, z_3 = -2, z_4 = 4$ and $w = \frac{3z-5}{z+1}$ (Answer: *Cross Ratio is invariant*)
2. $z_1 = i, z_2 = 3, z_3 = 2, z_4 = 0$ and $w = \frac{z+1}{z-1}$ (Answer: *Cross Ratio is invariant*)
3. $z_1 = 0, z_2 = 1, z_3 = -1, z_4 = i$ and $w = \frac{1+i-3z}{2-i+iz}$ (Answer: *Cross Ratio is invariant*)

Write a program to find the bilinear transformation which maps z_1, z_2, z_3 to w_1, w_2, w_3 respectively.

1. $z_1 = 1, z_2 = -1, z_3 = \infty$ and $w_1 = 1+i, w_2 = 1-i, w_3 = 1$ (Answer: $w = \frac{z+i}{z-i}$)
2. $z_1 = -i, z_2 = 0, z_3 = i$ and $w_1 = -1, w_2 = i, w_3 = 1$ (Answer: $w = -\frac{z(z-1)}{z+1}$)
3. $z_1 = -1, z_2 = 0, z_3 = 1$ and $w_1 = 0, w_2 = i, w_3 = 3i$ (Answer: $w = -\frac{3z(z+1)}{z-3}$)
4. $z_1 = 1, z_2 = i, z_3 = -1$ and $w_1 = i, w_2 = 0, w_3 = -i$ (Answer: $w = -\frac{z(iz+1)}{iz-1}$)

Experiment 6

Program to find fixed points of bilinear transformations.

Aim: To find the fixed points of a bilinear transformation using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
%i	Complex imaginary unit $\sqrt{-1}$
assume(r>0)	To declare that r is positive
realpart (expr)	Returns the real part of <i>expr</i> .
imagpart (expr)	Returns the imaginary part of the expression <i>expr</i> .
conjugate (z)	Returns the complex conjugate of <i>z</i> .
abs (z)	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
cabs (expr)	Calculates the absolute value of an expression representing a complex number.
atan2 (y, x)	yields the value of atan(<i>y/x</i>) in the interval $-\%pi$ to $\%pi$.
rectform (expr)	Returns an expression $a + b \%i$ equivalent to <i>expr</i> , such that <i>a</i> and <i>b</i> are purely real
:=	The function definition operator.
$f(x_1, \dots, x_n) := expr$	Defines a function named <i>f</i> with arguments $x_1, \dots, x_n$ and function body <i>expr</i>
subst ([eq_1, ..., eq_k], expr)	For each equation, the right side will be substituted for the left in the expression <i>expr</i> .
radcan (expr)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals.
if cond_1 then expr_1 else expr_0	evaluates to <i>expr_1</i> if <i>cond_1</i> evaluates to true, otherwise the expression evaluates to <i>expr_0</i> .
print ("text", expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
solve (expr, x)	Solves the algebraic equation <i>expr</i> for the variable <i>x</i> and returns a list of solution equations in <i>x</i>
inf	inf represents real positive infinity
%pi;	π , an irrational number

Note: 1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit)\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Bilinear transformation: A transformation of the form $w = \frac{az+b}{cz+d}$ is called a Bilinear transformation or linear fractional transformation, where a, b, c, d are complex constants and $ad - bc \neq 0$. This transformation is linear in both w and z and hence it is a bilinear transformation. It is also called Mobius transformation. Inverse transformation is $z = \frac{-dw+b}{cw-a}$

Fixed points or Invariant points of a transformation: If $w = f(z)$ is any transformation from the z -plane to the w -plane, the fixed points of the transformation are the solutions of the equation $z = f(z)$. i.e., if $w = z$ then it is a fixed point or invariant point of the transformation. A fixed point is mapped to itself in the transformation. Invariant points are got by taking $w = z$ in the transformation.

Fixed points or Invariant points of a Bilinear transformation: Fixed points of a bilinear transformation $w = \frac{az+b}{cz+d}$ are given by $z = \frac{az+b}{cz+d}$. On simplification this equation, we get $cz^2 + (d - a)z - b = 0$. Since, it is a quadratic in z , a bilinear transformation will have at most two fixed points. In fact,

1. $w = \frac{az+b}{cz+d}$ will have two finite fixed points if $c \neq 0$ and $(d - a)^2 + 4ac \neq 0$
2. $w = \frac{az+b}{cz+d}$ will have one finite fixed point if $c \neq 0$ and $(d - a)^2 + 4ac = 0$
3. $w = \frac{az+b}{cz+d}$ will have ∞ and one finite fixed point if $c = 0$ and $a \neq d$
4. $w = \frac{az+b}{cz+d}$ will have ∞ as the only fixed point if $c = 0$ and $a = d$

Program:

Program to find the fixed points of a given bilinear transformation.

```
w(z):= given function of z$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and", SOL[2])$
```


Worked Examples:

Problem 1. Write a program to find the fixed points of bilinear transformation $w = \frac{1-z}{1+z}$.

Program:

```
w(z):=(1-z)/(1+z)$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$
```

Output:

Given Bilinear Transformation is $w(z) = \frac{1-z}{z+1}$
Fixed points of $w(z)$ are $z = -\sqrt{2}-1$ and $z = \sqrt{2}-1$

```
w(z):=(1-z)/(1+z)$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$

Given Bilinear Transformation is w(z)=  $\frac{1-z}{z+1}$ 
Fixed points of w(z) are  $z = -\sqrt{2}-1$  and  $z = \sqrt{2}-1$ 
```

Problem 2. Write a program to find the fixed points of bilinear transformation $w = \frac{-3z+5i}{1-iz}$.

Program:

```
w(z):=(-3*z+5*i)/(1-%i*z)$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$
```

Output:

Given Bilinear Transformation is $w(z) = \frac{5\%i - 3z}{1 - \%iz}$
Fixed points of $w(z)$ are $z = \%i$ and $z = -5\%i$


```

w(z):=(-3-z+5*i)/(1-i*z)$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$

```

$$\text{Given Bilinear Transformation is } w(z) = \frac{5i - 3z}{1 - iz}$$

Fixed points of w(z) are $z = i$ and $z = -5i$

Problem 3. Write a program to find the fixed points of bilinear transformation $w = \frac{6z-9}{z}$

Program:

```

w(z):=(6*z-9)/z$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$

```

Output:

$$\text{Given Bilinear Transformation is } w(z) = \frac{6z - 9}{z}$$

Fixed point of (z) is $z = 3$

```

w(z):=(6-z-9)/z$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$

```

$$\text{Given Bilinear Transformation is } w(z) = \frac{6z - 9}{z}$$

Fixed point of w(z) is $z = 3$

Problem 4. Write a program to find the fixed points of bilinear transformation $w = z + 3$.

Program:

```

w(z):=(-3*z+5*i)/(1-i*z)$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$

```

```

if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$

```

Output:

Given Bilinear Transformation is $w(z) = z + 3$

Fixed point of $w(z)$ is ∞

```

w(z):=z+3$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$
Given Bilinear Transformation is w(z)= z + 3
Fixed point of w(z) is ∞

```

Problem 5. Write a program to find the fixed points of bilinear transformation $w = \frac{1}{z-2i}$.

Program:

```

w(z):=(-3*z+5*%i)/(1-%i*z)$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$

```

Output:

Given Bilinear Transformation is $w(z) = \frac{1}{z - 2i}$

Fixed point of $w(z)$ is $z = %i$

```

w(z):=1/(z-2*%i)$
SOL:solve(z=w(z),z)$
print("Given Bilinear Transformation is w(z)=",w(z))$
if length(SOL)=0 then print("Fixed point of w(z) is",inf)
elseif length(SOL)=1 then print("Fixed point of w(z) is",SOL[1])
else print("Fixed points of w(z) are",SOL[1], "and",SOL[2])$
Given Bilinear Transformation is w(z)=  $\frac{1}{z - 2i}$ 
Fixed point of w(z) is z=%i

```

Exercise:

Write a program to find the fixed points of given bilinear transformation w .

1. $w = \frac{3z-5}{z+1}$ (Answer: $z = 5i$ and $z = -5i$)

2. $w = \frac{6z-2i}{3-iz}$ (Answer: $z = 2i$ and $z = 0$)

3. $w = \frac{z+1}{1-z}$ (Answer: $z = -i$ and $z = i$)

4. $w = 2z + 3$ (Answer: $z = -3$)

5. $w = \frac{z}{2-z}$ (Answer: $z = 0$ and $z = 1$)

6. $w = \frac{3z-4}{z-1}$ (Answer: $z = 2$)

7. $w = \frac{3iz+1}{z+i}$ (Answer: $z = i$)

8. $w = \frac{(i+2)-2}{z+i}$ (Answer: $z = 1 - i$ and $z = i + 1$)

9. $w = \frac{z-3}{z+1}$ (Answer: $z = -\sqrt{3}i$ and $z = \sqrt{3}i$)

10. $w = \frac{1}{z}$ (Answer: $z = -1$ and $z = 1$)

Experiment 7

Program to verify De-Moivre's theorem.

Aim: To verify DeMoivre's theorem using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
%i	Complex imaginary unit $\sqrt{-1}$
assume(r>0)	To declare that r is positive
realpart (expr)	Returns the real part of <i>expr</i> .
imagpart (expr)	Returns the imaginary part of the expression <i>expr</i> .
conjugate (z)	Returns the complex conjugate of <i>z</i> .
abs (z)	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
cabs (expr)	Calculates the absolute value of an expression representing a complex number.
rectform (expr)	Returns an expression $a + b \%i$ equivalent to <i>expr</i> , such that a and b are purely real
atan2 (y, x)	yields the value of atan(y/x) in the interval $-\%pi$ to $\%pi$.
trigrat (expr)	Gives a canonical simplified quasilinear form of a trigonometrical expression; <i>expr</i> is a rational fraction of several sin, cos or tan, the arguments of them are linear forms in some variables (or kernels) and $\%pi/n$ (<i>n</i> integer) with integer coefficients.
subst (a, b, c)	Substitutes a for b in c
subst ([eq_1, ..., eq_k], expr)	For each equation, the right side will be substituted for the left in the expression <i>expr</i> .
radcan (expr)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals.
if cond_1 then expr_1 else expr_0	evaluates to <i>expr_1</i> if <i>cond_1</i> evaluates to true, otherwise the expression evaluates to <i>expr_0</i> .
exp (x) or %e^x	Represents the exponential function
sin (x)	Trigonometric function sine of x
%pi;	π , an irrational number

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit())\$ to remove previously assigned values of all symbols

Definitions and Formulae:

DeMoivre's Theorem: For any real number θ and integer n ,

$$(\cos(\theta) + i \sin(\theta))^n = \cos(n\theta) + i \sin(n\theta)$$

Further, for any real number θ and natural number n , $\cos\left(\frac{\theta}{n}\right) + i \sin\left(\frac{\theta}{n}\right)$ is one of roots of the n -th

roots of $(\cos(\theta) + i \sin(\theta))^{\frac{1}{n}}$ and all roots of n -th roots of $(\cos(\theta) + i \sin(\theta))^{\frac{1}{n}}$ are given by

$$\left(\cos\left(\frac{\theta + 2\pi k}{n}\right) + i \sin\left(\frac{\theta + 2\pi k}{n}\right)\right) \text{ for } k = 0, 1, 2, \dots, (n - 1).$$

In fact, for any rational number $\frac{p}{q}$ in canonical form (i.e., $p \in \mathbb{Z}$, $q \in \mathbb{N}$, $\text{HC}(p, q) = 1$), all

values of $(\cos(\theta) + i \sin(\theta))^{\frac{p}{q}}$ are given by

$$\left(\cos\left(\frac{p\theta + 2\pi k}{q}\right) + i \sin\left(\frac{p\theta + 2\pi k}{q}\right)\right) \text{ for } k = 0, 1, 2, \dots, (q - 1).$$

Verification of DeMoivre's Theorem: DeMoivre's theorem can be easily verified by equating its LHS with RHS for given values of θ and n .

Program:

Program to verify DeMoivre's Theorem for given values of n and θ

```
kill(all)$
n: given value of n (if given)$
θ: given value of θ (if given)$
LHS:(cos(θ)+%i*sin(θ))^n$
RHS:cos(n*θ)+%i*sin(n*θ)$
print("LHS=",LHS)$
print("On simplication, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$
```

NOTE: Sometimes we use `rectform(LHS)` in place of `trigrat(LHS)` in the above program for simplification. One can try different simplification functions to get right answer.

Worked Examples:

Problem 1. Program to verify DeMoivre's Theorem for $n = 2$ and any θ

Program:

```
kill(all)$
n:2$
LHS:(cos(θ)+%i*sin(θ))^n$
RHS:cos(n*θ)+%i*sin(n*θ)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$
```

```
kill(all)$
n:2$
LHS:(cos(θ)+%i*sin(θ))^n$
RHS:cos(n*θ)+%i*sin(n*θ)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$

LHS= (%i sin(θ)+cos(θ))2
On simplification, LHS= %i sin(2 θ)+cos(2 θ)
RHS= %i sin(2 θ)+cos(2 θ)
DeMoivre's Theorem is verified
```

Output:

$LHS = (%i \sin(\theta) + \cos(\theta))^2$
On simplification, LHS = %i sin(2θ) + cos(2θ)
 $RHS = %i \sin(2\theta) + \cos(2\theta)$
DeMoivre's Theorem is verified

Problem 2. Program to verify DeMoivre's Theorem for $n = -3$ and any θ

Program:

```
kill(all)$
n:-3$
LHS:(cos(θ)+%i*sin(θ))^n$
RHS:cos(n*θ)+%i*sin(n*θ)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$
```

```
kill(all)$
n:-3$
LHS:(cos(θ)+%i*sin(θ))^n$
RHS:cos(n*θ)+%i*sin(n*θ)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$

LHS=  $\frac{1}{(%i \sin(\theta) + \cos(\theta))^3}$ 
On simplification, LHS= cos(3 θ)-%i sin(3 θ)
RHS= cos(3 θ)-%i sin(3 θ)
DeMoivre's Theorem is verified
```

Output:

$$LHS = \frac{1}{(%i \sin(\theta) + \cos(\theta))^3}$$

On simplification, $LHS = \cos(3\theta) - \%i \sin(3\theta)$

$RHS = \cos(3\theta) - \%i \sin(3\theta)$

DeMoivre's Theorem is verified

Problem 3. Program to verify DeMoivre's Theorem for $n = \frac{1}{2}$ and any θ

Program:

```
kill(all)$
n:1/2$
LHS:(cos(theta)+%i*sin(theta))^n$
RHS:cos(n*theta)+%i*sin(n*theta)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$
```

```
kill(all)$
n:1/2$
LHS:(cos(theta)+%i*sin(theta))^n$
RHS:cos(n*theta)+%i*sin(n*theta)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$

LHS=  $\sqrt{\%i \sin(\theta) + \cos(\theta)}$ 
On simplification, LHS=  $\%i \sin\left(\frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)$ 
RHS=  $\%i \sin\left(\frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)$ 
DeMoivre's Theorem is verified
```

Output:

$LHS = \sqrt{\%i \sin(\theta) + \cos(\theta)}$

On simplification, $LHS = \%i \sin\left(\frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)$

$RHS = \%i \sin\left(\frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)$

DeMoivre's Theorem is verified

Problem 4. Program to verify DeMoivre's Theorem for $n = \frac{1}{3}$ and $\theta = \frac{\pi}{4}$

Program:

```
kill(all)$
n:1/3$
theta:pi/4$
LHS:(cos(theta)+%i*sin(theta))^n$
RHS:cos(n*theta)+%i*sin(n*theta)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
```

```
kill(all)$
n:1/3$
theta:pi/4$
LHS:(cos(theta)+%i*sin(theta))^n$
RHS:cos(n*theta)+%i*sin(n*theta)$
print("LHS=",LHS)$
print("On simplification, LHS=",trigrat(LHS))$
print("RHS=",RHS)$
if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$

LHS=  $\left(\frac{\%i}{\sqrt{2}} + \frac{1}{\sqrt{2}}\right)^{1/3}$ 
On simplification, LHS=  $\%i \sin\left(\frac{\pi}{12}\right) + \cos\left(\frac{\pi}{12}\right)$ 
RHS=  $\%i \sin\left(\frac{\pi}{12}\right) + \cos\left(\frac{\pi}{12}\right)$ 
DeMoivre's Theorem is verified
```

```

if trigrat(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$

```

Output:

$$LHS = \left(\frac{\%i}{\sqrt{2}} + \frac{1}{\sqrt{2}} \right)^{\frac{1}{3}}$$

On simplication, $LHS = \%i \sin\left(\frac{\pi}{12}\right) + \cos\left(\frac{\pi}{12}\right)$

$$RHS = \%i \sin\left(\frac{\pi}{12}\right) + \cos\left(\frac{\pi}{12}\right)$$

DeMoivre's Theorem is verified

Problem 5. Program to verify DeMoivre's Theorem for $n = \frac{-2}{3}$ and $\theta = \frac{\pi}{6}$

Program:

```

kill(all)$
n:-2/3$
theta:pi/6$
LHS:(cos(theta)+%i*sin(theta))^n$
RHS:cos(n*theta)+%i*sin(n*theta)$
print("LHS=",LHS)$
print("On simplication, LHS=",rectform(LHS))$
print("RHS=",RHS)$
if rectform(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$

```

```

kill(all)$
n:-2/3$
theta:pi/6$
LHS:(cos(theta)+%i*sin(theta))^n$
RHS:cos(n*theta)+%i*sin(n*theta)$
print("LHS=",LHS)$
print("On simplication, LHS=",rectform(LHS))$
print("RHS=",RHS)$
if rectform(LHS)=RHS then
print("DeMoivre's Theorem is verified")
else print("DeMoivre's Theorem is not verified")$

```

$$LHS = \frac{1}{\left(\frac{\%i}{2} + \frac{\sqrt{3}}{2} \right)^{2/3}}$$

On simplication, $LHS = \cos\left(\frac{\pi}{9}\right) - \%i \sin\left(\frac{\pi}{9}\right)$

$$RHS = \cos\left(\frac{\pi}{9}\right) - \%i \sin\left(\frac{\pi}{9}\right)$$

DeMoivre's Theorem is verified

Output:

$$LHS = \frac{1}{\left(\frac{\%i}{2} + \frac{\sqrt{3}}{2} \right)^{\frac{2}{3}}}$$

On simplication, $LHS = \cos\left(\frac{\pi}{9}\right) - \%i \sin\left(\frac{\pi}{9}\right)$

$$RHS = \cos\left(\frac{\pi}{9}\right) - \%i \sin\left(\frac{\pi}{9}\right)$$

DeMoivre's Theorem is verified

Exercise:

Write a program to verify DeMoivre's Theorem for given n and θ :

1. $n = \frac{1}{100}$ and any θ (Answer: *DeMoivre's Theorem is verified*)

2. $n = -3$ and $\theta = \frac{\pi}{6}$ (Answer: *DeMoivre's Theorem is verified*)

3. $n = \frac{1}{10}$ and $\theta = \frac{\pi}{3}$ (Answer: *DeMoivre's Theorem is verified*)

4. $n = -4$ and $\theta = \frac{\pi}{3}$ (Answer: *DeMoivre's Theorem is verified*)

5. $n = -3$ and $\theta = 3$ (Answer: *DeMoivre's Theorem is verified*)

6. $n = 4$ and $\theta = 4$ (Answer: *DeMoivre's Theorem is verified*)

7. $n = \frac{-2}{3}$ and $\theta = \frac{-\pi}{4}$ (Answer: *DeMoivre's Theorem is verified*)

Experiment 9

Program to find upper and the lower Riemann sums with respect to a given partition.

If $a = x_0 < x_1 < \dots < x_n = b$, then the finite ordered set $P = \{x_0, x_1, \dots, x_n\}$ is called a partition of $[a, b]$.

Let $f: [a, b] \rightarrow \mathbb{R}$ be a bounded function and $P = \{a = x_0, x_1, \dots, x_n = b\}$ be a partition of $[a, b]$. Let $\delta_r = x_r - x_{r-1}$, is the length of each subinterval $[x_r, x_{r-1}]$, $r = 1, 2, \dots, n$. f is bounded on $[a, b] \Rightarrow f$ is bounded on each subintervals $[x_r, x_{r-1}]$, $r = 1, 2, \dots, n$. Let $M_r = \text{Supremum of } f \text{ and } m_r = \text{Infimum of } f \text{ in } [x_r, x_{r-1}]$, $r = 1, 2, \dots, n$. Then $L(P, f) = \sum_{r=1}^n m_r \delta_r$ is called the lower Reimann sum of f and $U(P, f) = \sum_{r=1}^n M_r \delta_r$ is called the upper Reimann sum of f .

Program: Write a Maxima program to find upper and the lower Riemann sum of $f(x) = x^2$ over the partition $p = \{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\}$ of $[0, 1]$.

```
kill(all)$
f(x):=x^2;
a:0$
b:1$
n:4$
delta_x:(b-a)/n$
p:makelist(a+i*delta_x,i,0,n);
l_sum:0$
u_sum:0$
for i:1 thru length(p)-1 do
(
i1(i):=[p[i],p[i+1]],
d(i):=p[i+1]-p[i],
m(i):=f(p[i]),
M(i):=f(p[i+1]),
l_sum:l_sum+m(i)*d(i),
u_sum:u_sum+M(i)*d(i));
print("Upper sum",u_sum)$
print("Lower sum",l_sum)$
```

Output:

```
(%o1) f(x):=x^2
(%o6) [0,1/4,1/2,3/4,1]
(%o9) done
```

"Upper sum" "15/32"

"Lower sum" "7/32"

Exercise:

Write a Maxima program to find upper and the lower Riemann sum of

- $f(x) = \sin(x)$ for $x \in [0, \frac{\pi}{2}]$ and $p = \{0, \frac{\pi}{4}, \frac{\pi}{2}\}$ of $[0, \pi]$.
- $f(x) = x^3$ for $x \in [0, 1]$ and $p = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$ of $[0, 1]$.

```
kill(all)$
f(x):=x^2;
a:0$
b:1$
n:4$
delta_x:(b-a)/n$
p:makelist(a+i*delta_x,i,0,n);
l_sum:0$
u_sum:0$
for i:1 thru length(p)-1 do
(
i1(i):=[p[i],p[i+1]],
d(i):=p[i+1]-p[i],
m(i):=f(p[i]),
M(i):=f(p[i+1]),
l_sum:l_sum+m(i)*d(i),
u_sum:u_sum+M(i)*d(i));
print("Upper sum",u_sum)$
print("Lower sum",l_sum)$
```

$$f(x) := x^2$$

$$\left[0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1 \right]$$

done

$$\text{Upper sum } \frac{15}{32}$$

$$\text{Lower sum } \frac{7}{32}$$

Experiment 10

Program to test Riemann integrability.

Lower Riemann Integral of f on $[a,b]$ is defined as $\sup \{L(P,f) \mid P \in \mathcal{P}[a,b]\}$ and is denoted by $\int_a^b f(x)dx$.

Upper Riemann Integral of f on $[a,b]$ is defined as $\inf \{U(P,f) \mid P \in \mathcal{P}[a,b]\}$ and is denoted by $\int_a^b f(x)dx$.

Riemann Integral – A bounded function f is said to be Riemann integrable on $[a,b]$ if

$$\int_a^b f(x)dx = \int_a^b f(x)dx \text{ and the common value is denoted by } \int_a^b f(x)dx$$

NOTE: Let $P=\{a=\}$ be a partition of $[a,b]$ with n equal subintervals. Then length of each subinterval is $\frac{(b-a)}{n}$.

If $f(x)$ is monotonically increasing in the subinterval $[x_r, x_{r-1}]$, then maximum attains at x_r and hence $M_r = \text{Supremum of } f = f(x_r) = f\left(a + r \frac{b-a}{n}\right)$, minimum attains at x_{r-1} and hence $m_r = \text{infimum of } f = f(x_{r-1}) = f\left(a + (r-1) \frac{b-a}{n}\right)$.

If $f(x)$ is monotonically decreasing in the subinterval $[x_r, x_{r-1}]$, then maximum attains at x_{r-1} and hence $M_r = \text{Supremum of } f = f(x_{r-1}) = f\left(a + (r-1) \frac{b-a}{n}\right)$, minimum attains at x_r and hence $m_r = \text{infimum of } f = f(x_r) = f\left(a + r \frac{b-a}{n}\right)$.

Program: Write a Maxima program to verify the function $f(x) = x^3$ is Riemann integral or not over $[0, 1]$.

```
kill(all)$
f(x):=x^3;
a:0$
b:1$
lr:[(r-1)*b)/n,(r*b)/n]$
Mr:f(lr[2])$
mr:f(lr[1])$
dr:ratsimp(lr[2]-lr[1])$
Ur:sum((mr*dr),r,1,n),simpsum$
Ur1:ratsimp(Ur)$
Lr:sum((mr*dr),r,1,n),simpsum$
Lr1:ratsimp(Lr)$
U_rsum:limit(Ur1,n,inf)$
print("Upper Riemann integral is",U_rsum)$
L_rsum:limit(Lr1,n,inf)$
print("Lower Riemann integral is",L_rsum)$
if U_rsum = L_rsum then
print("the given function is R-integrable on", [a,b])
else
print("the given function is not R-integrable on", [a,
```

Output:

```
(%o1) f(x):=x^3
```

```
"Upper Riemann integral is" "1/4" "
```

```
"Lower Riemann integral is" "1/4" "
```

```
"the given function is R-integrable on [0,1]" "
```

```
kill(all)$
f(x):=x^3;
a:0$
b:1$
lr:[(r-1)*b)/n,(r*b)/n]$
Mr:f(lr[2])$
mr:f(lr[1])$
dr:ratsimp(lr[2]-lr[1])$
Ur:sum((mr*dr),r,1,n),simpsum$
Ur1:ratsimp(Ur)$
Lr:sum((mr*dr),r,1,n),simpsum$
Lr1:ratsimp(Lr)$
U_rsum:limit(Ur1,n,inf)$
print("Upper Riemann integral is",U_rsum)$
L_rsum:limit(Lr1,n,inf)$
print("Lower Riemann integral is",L_rsum)$
if U_rsum = L_rsum then
print("the given function is R-integrable on", [a, b])
else
print("the given function is not R-integrable on", [a,b])$
```

$$f(x) := x^3$$

$$\text{Upper Riemann integral is } \frac{1}{4}$$

$$\text{Lower Riemann integral is } \frac{1}{4}$$

the given function is R-integrable on $[0, 1]$

Exercise:

Write a Maxima program to verify the function

1. $f(x) = 3x + 1$ is Reimann integral or not over $[1, 3]$.

Experiment 11

Program to evaluate Riemann integral as a limit of sum.

Program: Write a Maxima program to evaluate Riemann integral as a limit of sum of $f(x) = 2x^2 - 3x + 5$ in $[0, 1]$.

```
kill(all)$
f(x):=2*x^2-3*x+5;
print("The function is bounded and continuous and hence integrable")$
a:0$
b:1$
dr:(b-a)/n$
xr:a+r*dr$
sum:sum(f(xr)*dr,r,1,n),simpsum$
s:ratsimp(sum)$
L_sum:limit(s,n,inf)$
print("Riemann integral as limit of sum is",L_sum)$
```

Output:

(%o1) $f(x) = 2x^2 - 3x + 5$

"The function is bounded and continuous and hence integrable"
" "

"Riemann integral as limit of sum is" "25/6" "

```
kill(all)$
f(x):=2*x^2-3*x+5;
print("The function is bounded and continuous and hence integrable")$
a:0$
b:1$
dr:(b-a)/n$
xr:a+r*dr$
sum:sum(f(xr)*dr,r,1,n),simpsum$
s:ratsimp(sum)$
L_sum:limit(s,n,inf)$
print("Riemann integral as limit of sum is",L_sum)$
```

$$f(x) = 2x^2 - 3x + 5$$

The function is bounded and continuous and hence integrable

Riemann integral as limit of sum is $\frac{25}{6}$

Experiment 12

Program to evaluate $\Gamma(n)$ for n is positive integer and non-integer.

If $n > 0$ then the integral $\int_0^{\infty} x^{n-1} e^{-x} dx$ which is obviously a function of n , is called a Gamma function of n and is denoted by $\Gamma(n)$. Thus $\Gamma(n) = \int_0^{\infty} x^{n-1} e^{-x} dx \forall n > 0$. Gamma function is called the Second Eulerian Integral.

Program: Write a maxima program to evaluate $\Gamma(2)$.

```
kill(all)$  
gamma(2);
```

Output:

1

Program: Write a maxima program to evaluate $\Gamma(1/2)$.

```
kill(all)$  
gamma(1/2);
```

Output:

```
sqrt(%pi)
```

Exercise:

Write a program to evaluate

- | | |
|---------------------|-----------------------------|
| 1. $\Gamma(5)$. | Ans: 24 |
| 2. $\Gamma(7)$. | Ans: 720 |
| 3. $\Gamma(-1/2)$. | Ans: $-2\sqrt{\pi}$ |
| 4. $\Gamma(3/2)$. | Ans: $\frac{\sqrt{\pi}}{2}$ |

Experiment 13

Program to evaluate $\beta(m, n)$ for $m > 0$ and $n > 0$.

If $m > 0, n > 0$ then the integral $\int_0^1 x^{m-1}(1-x)^{n-1} dx$ which is obviously a function of m and n is called a Beta function and is denoted by $\beta(m, n)$. Thus $\beta(m, n) = \int_0^1 x^{m-1}(1-x)^{n-1} dx \forall m > 0, n > 0$. Beta function is called the first Eulerian Integral.

Program: Write a maxima program to evaluate $\beta(x, y)$.

```
kill(all)$  
makefact(beta(x,y));
```

Output:

```
((x-1)!*(y-1)!)/(y+x-1)!
```

Program: Write a maxima program to evaluate $\beta(2,3)$.

```
kill(all)$  
makefact(beta(2,3));
```

Output:

```
1/12
```

Exercise:

Write a program to evaluate

- | | |
|--|--------------------------|
| 1. $\beta\left(\frac{5}{2}, \frac{3}{2}\right)$ | Ans: $\frac{\pi}{16}$ |
| 2. $\beta\left(\frac{1}{2}, \frac{11}{2}\right)$ | Ans: $\frac{63\pi}{256}$ |



BLDEA'S S.B. ARTS AND K.C.P. SCIENCE

COLLEGE, VIJAYAPUR

(Affiliated to Rani Channamma University, Belagavi)

Practical's on DSC 2

**VECTOR CALCULUS AND ANALYTICAL
GEOMETRY**

(AS PER NEP-2020)

MANUAL

B. Sc. V Semester

Name:

Class:

UUCMS No:

DEPARTMENT OF MATHEMATICS

CONTENTS		
Sl. No.	Title	Page No
1	Contents	2
2	List of Experiments	3
3	Experiment 1 Program on multiple product of vectors: Scalar and Cross product	4
4	Experiment 2 Program on vector differentiation and finding unit tangent.	12
5	Experiment 3 Program to find curvature and torsion of a space curve.	18
6	Experiment 4 Program to find the gradient and the Laplacian of a scalar function and divergence and curl of a vector function.	25
7	Experiment 5 Program to demonstrate the physical interpretation of gradient, divergence and curl.	33
8	Experiment 6 Program to evaluate a vector line integral.	41
9	Experiment 7 Program to evaluate a surface integral.	43
10	Experiment 8 Program to evaluate a volume integral.	45
11	Experiment 9 Program to verify Green's theorem.	47
12	Experiment 10 Program to find equation and plot sphere, cone and cylinder.	50
13	Experiment 11 Program to find distance between a straight line and a plane.	53
14	Experiment 12 Program to construct and plot some standard surfaces.	56

List of Experiments

For Fifth Semester DSC 2 Mathematics
(Practical on Vector Calculus and Analytical Geometry)

(4 Hours per Week and 56 hours per Semester)

1. Program on multiple product of vectors: Scalar and Cross product
2. Program on vector differentiation and finding unit tangent.
3. Program to find curvature and torsion of a space curve.
4. Program to find the gradient and the Laplacian of a scalar function and divergence and curl of a vector function.
5. Program to demonstrate the physical interpretation of gradient, divergence and curl.
6. Program to evaluate a vector line integral.
7. Program to evaluate a surface integral.
8. Program to evaluate a volume integral.
9. Program to verify Green's theorem.
10. Program to find equation and plot sphere, cone and cylinder.
11. Program to find distance between a straight line and a plane.
12. Program to construct and plot some standard surfaces.

Experiment 1

Program on multiple product of vectors: Scalar and Cross product.

Aim: To find the Scalar and Cross Product of two vectors and Scalar triple product and Vector triple product of three vectors using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
load ("vect")	vect is a package of functions for vector analysis. load ("vect") loads this package
<i>express (expr)</i>	Expands differential operator nouns into expressions in terms of partial derivatives. <i>express</i> recognizes the operators <i>grad</i> , <i>div</i> , <i>curl</i> , <i>laplacian</i> .
:=	To define a function/expression
<i>diff</i>	When <i>diff</i> is present as an <i>evflag</i> in call to <i>ev</i> , all differentiations indicated in <i>expr</i> are carried out.
grad()	gradient operator
div()	divergence operator
laplacian()	Laplacian operator
curl()	curl operator
~	The wedge product operator is denoted by the tilde ~. This is used to compute cross product of vectors.
* (asterik)	Commutative Multiplication
. (dot)	Noncommutative multiplication and scalar product
coeff (expr, x)	Returns the coefficient of <i>x</i> in <i>expr</i> , where <i>expr</i> is a polynomial or a monomial term in <i>x</i> .
apply('matrix, nested lists)	Converts nested lists of same length into a matrix
[a_1,...,a_n];	To create a list [a_1,...,a_n]
A[i]	[and] also enclose the subscripts of a list. A[i] will be i-th element of list A
expand (expr)	Expands expression <i>expr</i> .
^ (Carrot Symbol) or **	For index/power/exponentiation (Commutative)
trigreduce (expr, x)	Combines products and powers of trigonometric and hyperbolic sin's and cos's of <i>x</i> into those of multiples of <i>x</i> . It also tries to eliminate these functions when they occur in denominators. If <i>x</i> is omitted then all variables in <i>expr</i> are used.
determinant (M)	Computes the determinant of <i>M</i>
print ("text", expr)\$	Displays text within inverted commas and evaluates and displays <i>expr</i>
radcan(expr)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals
acos()	arc cos or $\cos^{-1}$ function
abs ()	The abs function represents the mathematical absolute value function and works for both numerical and symbolic values.
mat_norm (M, frobenius)	Return the frobenius (the Frobenius matrix norm) of the matrix <i>M</i> .
load ("eigen")	The package eigen contains several functions devoted to the symbolic computation of eigenvalues and eigenvectors. load ("eigen") loads this package
unitvector (x)	Returns $x/\text{norm}(x)$; this is a unit vector in the same direction as <i>x</i> . load ("eigen") loads this function.

Definitions and Formulae:

Plane Vectors and Space Vectors: Let $P(x, y)$ be a point in the plane. The position vector of P is given by $\vec{r} = x \hat{i} + y \hat{j}$ where $\hat{i}$ and $\hat{j}$ are unit vectors along positive x -axis and y -axis respectively. Similarly, for a point $P(x, y, z)$ in the space, the position vector is $\vec{r} = x \hat{i} + y \hat{j} + z \hat{k}$ where $\hat{i}$ and $\hat{j}$ and $\hat{k}$ are unit vectors along positive x -axis, y -axis and z -axis respectively. Thus, a vector $\vec{a} = x \hat{i} + y \hat{j}$ where $x, y \in \mathbb{R}$ is a plane vector and $\vec{a} = x \hat{i} + y \hat{j} + z \hat{k}$ where $x, y, z \in \mathbb{R}$ is a space vector.

Magnitude of a vector: Magnitude of a plane vector $\vec{a} = x \hat{i} + y \hat{j}$ is $|\vec{a}| = \sqrt{x^2 + y^2}$ and magnitude of a space vector $\vec{a} = x \hat{i} + y \hat{j} + z \hat{k}$ is $|\vec{a}| = \sqrt{x^2 + y^2 + z^2}$.

Unit vector: Unit vector in the direction of $\vec{a}$ is $\hat{a} = \frac{\vec{a}}{|\vec{a}|}$

Scalar Product/ Dot product of two space vectors: For two space vectors $\vec{a} = x_1 \hat{i} + y_1 \hat{j} + z_1 \hat{k}$ and $\vec{b} = x_2 \hat{i} + y_2 \hat{j} + z_2 \hat{k}$ their scalar product or dot product is the scalar $\vec{a} \cdot \vec{b} = x_1 x_2 + y_1 y_2 + z_1 z_2$. Also note that $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$.

Angle two space vectors: Angle between two space vectors $\vec{a} = x_1 \hat{i} + y_1 \hat{j} + z_1 \hat{k}$ and $\vec{b} = x_2 \hat{i} + y_2 \hat{j} + z_2 \hat{k}$ is given by $\theta = \cos^{-1} \left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} \right)$.

Vector Product/ Cross Product of two space vectors: For two space vectors $\vec{a} = x_1 \hat{i} + y_1 \hat{j} + z_1 \hat{k}$ and $\vec{b} = x_2 \hat{i} + y_2 \hat{j} + z_2 \hat{k}$ their cross product or vector product $\vec{a} \times \vec{b}$ is the vector given by

$$\vec{a} \times \vec{b} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} \text{ and } \vec{b} \times \vec{a} = -\vec{a} \times \vec{b}$$

Scalar Triple Product/ Box product of three space vectors: For three space vectors $\vec{a} = x_1 \hat{i} + y_1 \hat{j} + z_1 \hat{k}$, $\vec{b} = x_2 \hat{i} + y_2 \hat{j} + z_2 \hat{k}$ and $\vec{c} = x_3 \hat{i} + y_3 \hat{j} + z_3 \hat{k}$ their scalar triple product or box product

$$(\vec{a} \times \vec{b}) \cdot \vec{c} = \vec{a} \cdot (\vec{b} \times \vec{c}) \text{ is the scalar } [\vec{a}, \vec{b}, \vec{c}] = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}.$$

Vector Triple Product of three space vectors: For three space vectors $\vec{a} = x_1 \hat{i} + y_1 \hat{j} + z_1 \hat{k}$, $\vec{b} = x_2 \hat{i} + y_2 \hat{j} + z_2 \hat{k}$ and $\vec{c} = x_3 \hat{i} + y_3 \hat{j} + z_3 \hat{k}$ their vector triple products are the vectors given by:

$$(\vec{a} \times \vec{b}) \times \vec{c} = (\vec{a} \cdot \vec{c}) \vec{b} - (\vec{b} \cdot \vec{c}) \vec{a}$$

$$\vec{a} \times (\vec{b} \times \vec{c}) = (\vec{a} \cdot \vec{c}) \vec{b} - (\vec{a} \cdot \vec{b}) \vec{c}$$

Area of Parallelogram and Volume of a Parallelepiped: Area of parallelogram whose sides are vectors $\vec{a}$ and $\vec{b}$ is $|\vec{a} \times \vec{b}|$ and the volume of a parallelepiped whose edges are vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ is the absolute value of their scalar triple product $[\vec{a}, \vec{b}, \vec{c}]$.

Program:

Program to find the scalar product and cross product of two vectors. Also, to find the magnitude.

(i.e. to find $\vec{a} \cdot \vec{b}$, $\vec{a} \times \vec{b}$, $\vec{b} \times \vec{a}$, $|\vec{a} \times \vec{b}|$ and so on for $\vec{a} = a_1 \hat{i} + a_2 \hat{j} + a_3 \hat{k}$ and $\vec{b} = b_1 \hat{i} + b_2 \hat{j} + b_3 \hat{k}$)

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
dot(x,y):=x.y$
cross(x,y):=ev(express(x~y),diff)$
J:[i,j,k]$
a:[a1, a2, a3] $
b: [b1, b2, b3] $
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Dot Product a.b=",dot(a,b))$
print("Cross Product axb=", cross(a,b).J)$
print("Cross Product bxa=", cross(b,a).J)$
print("Magnitude of axb=", norm(cross(a,b)))$
```

Program to find the scalar triple product and vector triple product of three vectors. (i.e. to find $[\vec{a}, \vec{b}, \vec{c}]$, $(\vec{a} \times \vec{b}) \times \vec{c}$, $\vec{a} \times (\vec{b} \times \vec{c})$ and so on for $\vec{a} = a_1 \hat{i} + a_2 \hat{j} + a_3 \hat{k}$, $\vec{b} = b_1 \hat{i} + b_2 \hat{j} + b_3 \hat{k}$ and $\vec{c} = c_1 \hat{i} + c_2 \hat{j} + c_3 \hat{k}$)

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
vtp1(x,y,z):=ev(express(cross(x,y)~z),diff)$
vtp2(x,y,z):=ev(express(x~cross(y,z)),diff)$
J:[i,j,k]$
a:[a1, a2, a3] $
b: [b1, b2, b3] $
c:[c1, c2, c3]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("vector c=", c.J)$
print("Scalar triple product [a,b,c]=", stp(a,b,c))$
print("Vector triple product (axb)xc=", vtp1(a,b,c).J)$
print("Vector triple product ax(bxc)=", vtp2(a,b,c).J)$
print("Magnitude of (axb)xc=", norm(vtp1(a,b,c)))$
print("Magnitude of ax(bxc)=", norm(vtp2(a,b,c)))$
```

Worked Examples:

Problem 1. Write a program to find i) $\vec{a} \cdot \vec{b}$ ii) $\vec{a} \times \vec{b}$ iii) $\vec{b} \times \vec{a}$ and iv) $|\vec{a} \times \vec{b}|$
 where $\vec{a} = 2\hat{i} - 3\hat{j} - \hat{k}$ and $\vec{b} = \hat{i} + 4\hat{j} - 2\hat{k}$

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
dot(x,y):=x.y$
cross(x,y):=ev(express(x~y),diff)$
J:[i,j,k]$
a:[2,-3,-1]$
b:[1,4,-2]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Dot Product a.b=",dot(a,b))$
print("Cross Product axb=", cross(a,b).J)$
print("Cross Product bxa=", cross(b,a).J)$
print("Magnitude of axb=", norm(cross(a,b)))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
dot(x,y):=x.y$
cross(x,y):=ev(express(x~y),diff)$
J:[i,j,k]$
a:[2,-3,-1]$
b:[1,4,-2]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Dot Product a.b=",dot(a,b))$
print("Cross Product axb=", cross(a,b).J)$
print("Cross Product bxa=", cross(b,a).J)$
print("Magnitude of axb=", norm(cross(a,b)))$
vector a= -k -3 j +2 i
vector b= -2 k +4 j +i
Dot Product a.b= -8
Cross Product axb= 11 k +3 j +10 i
Cross Product bxa= -11 k -3 j -10 i
Magnitude of axb= sqrt(230)
```

Output:

```
vector a = -k - 3j + 2i
vector b = -2k + 4j + i
Dot Product a.b = -8
Cross Product axb = 11k + 3j + 10i
Cross Product bxa = -11k - 3j - 10i
Magnitude of axb = sqrt(230)
```

Problem 2. Write a program to find i) $\vec{a} \cdot \vec{b}$ ii) $\vec{a} \times \vec{b}$ iii) $|\vec{a} \times \vec{b}|$ iv) $(\vec{a} + \vec{b}) \times (\vec{a} - \vec{b})$
 v) unit vector of $\vec{a}$ where $\vec{a} = 2\hat{i} + 3\hat{j} - \hat{k}$ and $\vec{b} = \hat{i} - \hat{j} - 2\hat{k}$

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
dot(x,y):=x.y$
cross(x,y):=ev(express(x~y),diff)$
J:[i,j,k]$
a:[2,3,-1]$
b:[1,-1,-2]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Dot Product a.b=",dot(a,b))$
print("Cross Product axb=", cross(a,b).J)$
print("Magnitude of axb=", norm(cross(a,b)))$
print("Cross Product (a+b)x(a-b)=", cross(a+b,a-b).J)$
print("Unit vector of a=", expand(a.J/norm(a)))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
dot(x,y):=x.y$
cross(x,y):=ev(express(x~y),diff)$
J:[i,j,k]$
a:[2,3,-1]$
b:[1,-1,-2]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Dot Product a.b=",dot(a,b))$
print("Cross Product axb=", cross(a,b).J)$
print("Magnitude of axb=", norm(cross(a,b)))$
print("Cross Product (a+b)x(a-b)=", cross(a+b,a-b).J)$
print("Unit vector of a=", expand(a.J/norm(a)))$
vect: warning: removing existing rule or rules for ".":
vector a= -k +3 j +2 i
vector b= -2 k -j +i
Dot Product a.b= 1
Cross Product axb= -5 k +3 j -7 i
Magnitude of axb= sqrt(83)
Cross Product (a+b)x(a-b)= 10 k -6 j +14 i
Unit vector of a= -k/sqrt(14) + 3j/sqrt(14) + 2i/sqrt(14)
```

Output:

$$\begin{aligned} \text{vector } a &= -k + 3j + 2i \\ \text{vector } b &= -2k - j + i \\ \text{Dot Product } a \cdot b &= 1 \\ \text{Cross Product } a \times b &= -5k + 3j - 7i \\ \text{Magnitude of } a \times b &= \sqrt{83} \\ \text{Cross Product } (a + b) \times (a - b) &= 10k - 6j + 14i \\ \text{Unit vector of } a &= -\frac{k}{\sqrt{14}} + \frac{3j}{\sqrt{14}} + \frac{2i}{\sqrt{14}} \end{aligned}$$

Problem 3. Write a program to find i) $[\vec{a}, \vec{b}, \vec{c}]$ ii) $\vec{a} \times (\vec{b} \times \vec{c})$ iii) $\vec{a} \times (\vec{b} \times \vec{c})$ iv) $|\vec{a} \times (\vec{b} \times \vec{c})|$ v) $|\vec{a} \times (\vec{b} \times \vec{c})|$ where $\vec{a} = 3\hat{i} - \hat{j} + 2\hat{k}$ $\vec{b} = 2\hat{i} + \hat{j} - \hat{k}$ and $\vec{c} = \hat{i} - 2\hat{j} + 2\hat{k}$

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
vtp1(x,y,z):=ev(express(cross(x,y)~z),diff)$
vtp2(x,y,z):=ev(express(x~cross(y,z)),diff)$
J:[i,j,k]$
a:[3,-1,2]$
b:[2,1,-1]$
c:[1,-2,2]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("vector c=", c.J)$
print("Scalar triple product [a,b,c]=", stp(a,b,c))$
print("Vector triple product (axb)xc=", vtp1(a,b,c).J)$
print("Vector triple product ax(bxc)=", vtp2(a,b,c).J)$
print("Magnitude of (axb)xc=", norm(vtp1(a,b,c)))$
print("Magnitude of ax(bxc)=", norm(vtp2(a,b,c)))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
vtp1(x,y,z):=ev(express(cross(x,y)~z),diff)$
vtp2(x,y,z):=ev(express(x~cross(y,z)),diff)$
J:[i,j,k]$
a:[3,-1,2]$
b:[2,1,-1]$
c:[1,-2,2]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("vector c=", c.J)$
print("Scalar triple product [a,b,c]=", stp(a,b,c))$
print("Vector triple product (axb)xc=", vtp1(a,b,c).J)$
print("Vector triple product ax(bxc)=", vtp2(a,b,c).J)$
print("Magnitude of (axb)xc=", norm(vtp1(a,b,c)))$
print("Magnitude of ax(bxc)=", norm(vtp2(a,b,c)))$
vect: warning: removing existing rule or rules for "."
vector a= 2 k - j + 3 i
vector b= - k + j + 2 i
vector c= 2 k - 2 j + i
Scalar triple product [a,b,c]= -5
Vector triple product (axb)xc= -5 k + 7 j + 24 i
Vector triple product ax(bxc)= -15 k + 15 j + 15 i
Magnitude of (axb)xc= 5  $\sqrt{26}$ 
Magnitude of ax(bxc)= 5  $3^{1/2}$ 
```

Output:

$$\begin{aligned} \text{vector } a &= 2k - j + 3i \\ \text{vector } b &= -k + j + 2i \\ \text{vector } c &= 2k - 2j + i \\ \text{Scalar triple product } [a, b, c] &= -5 \\ \text{Vector triple product } (axb)xc &= -5k + 7j + 24i \\ \text{Vector triple product } ax(bxc) &= -15k + 15j + 15i \end{aligned}$$

$$\text{Magnitude of } (axb) \times c = 5\sqrt{26}$$

$$\text{Magnitude of } ax(b \times c) = 53^{\frac{3}{2}}$$

Note: In the above answer $53^{\frac{3}{2}} = 5 \times 3^{\frac{3}{2}} = 5\sqrt{3^3} = 15\sqrt{3}$. Don't confuse it as $(53)^{\frac{3}{2}}$

Problem 4. Write a program to find the angle between $\vec{a} = 2\hat{i} + 3\hat{j} + \hat{k}$ and $\vec{b} = 4\hat{i} - 2\hat{j} - 2\hat{k}$

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
dot(x,y):=x.y$
J:[i,j,k]$
a:[2,3,1]$
b:[4,-2,-2]$
theta:acos(dot(a,b)/(norm(a)*norm(b)))$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Angle between a and b is theta=", theta)$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
dot(x,y):=x.y$
J:[i,j,k]$
a:[2,3,1]$
b:[4,-2,-2]$
theta:acos(dot(a,b)/(norm(a)*norm(b)))$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Angle between a and b is theta=", theta)$
vector a= k+3j+2i
vector b= -2k-2j+4i
Angle between a and b is theta= \frac{\pi}{2}
```

Output:

$vector\ a = k + 3j + 2i$
 $vector\ b = -2k - 2j + 4i$
 $Angle\ between\ a\ and\ b\ is\ \theta = \frac{\pi}{2}$

Problem 5. Write a program to find the area of parallelogram whose sides are

$$\vec{a} = \hat{i} - 4\hat{j} - \hat{k} \text{ and } \vec{b} = -2\hat{i} - \hat{j} + \hat{k}$$

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
J:[i,j,k]$
a:[1,-4,-1]$
b:[-2,-1,1]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Area of Parallelogram=", norm(cross(a,b)))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
J:[i,j,k]$
a:[1,-4,-1]$
b:[-2,-1,1]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("Area of Parallelogram=", norm(cross(a,b)))$
vector a= -k-4j+i
vector b= k-j-2i
Area of Parallelogram= \sqrt{107}
```


Output:

$$\text{vector } a = k + 3j + 2i$$

$$\text{vector } b = -2k - 2j + 4i$$

$$\text{Area of Parallelogram} = \sqrt{107}$$

Problem 6. Write a program to find the volume of parallelepiped whose edges are

$$\vec{a} = 2\hat{i} - 3\hat{j}, \vec{b} = \hat{i} + \hat{j} - \hat{k} \text{ and } \vec{c} = 3\hat{i} - \hat{k}$$

Program:

```
load("vect")$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
a:[2,-3,0]$
b:[1,1,-1]$
c:[3,0,-1]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("vector c=", c.J)$
print("Volume of Parallelopiped =", abs(stp(a,b,c)))$
```

```
load("vect")$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
a:[2,-3,0]$
b:[1,1,-1]$
c:[3,0,-1]$
print("vector a=", a.J)$
print("vector b=", b.J)$
print("vector c=", c.J)$
print("Volume of Parallelopiped =", abs(stp(a,b,c)))$
vector a= 2 i - 3 j
vector b= -k + j + i
vector c= 3 i - k
Volume of Parallelopiped = 4
```

Output:

$$\text{vector } a = 2i - 3j$$

$$\text{vector } b = -k + j + i$$

$$\text{vector } c = 3i - k$$

$$\text{Volume of Parallelepiped} = 4$$

Exercise:

I. Write a program to find i) $\vec{a} \cdot \vec{b}$ ii) $\vec{b} \cdot \vec{a}$ iii) $\vec{a} \times \vec{b}$ iv) $\vec{b} \times \vec{a}$ v) $|\vec{a} \times \vec{b}|$ vi) $|\vec{b} \times \vec{a}|$
vii) $|\vec{a}|$ viii) $|\vec{b}|$ for given pair of vectors:

1. $\vec{a} = 2\hat{i} + 2\hat{j} - \hat{k}$ and $\vec{b} = 6\hat{i} - 3\hat{j} + 2\hat{k}$

2. $\vec{a} = 3\hat{i} - 2\hat{j} + \hat{k}$ and $\vec{b} = \hat{i} - 3\hat{j} + 5\hat{k}$

II. Write a program to find i) $[\vec{a}, \vec{b}, \vec{c}]$ ii) $\vec{a} \times (\vec{b} \times \vec{c})$ iii) $\vec{b} \times (\vec{a} \times \vec{c})$ for the following space vectors:

1. $\vec{a} = \hat{i} - 2\hat{j} - 3\hat{k}$ $\vec{b} = 2\hat{i} + \hat{j} - \hat{k}$ and $\vec{c} = \hat{i} + 3\hat{j} - 2\hat{k}$

2. $\vec{a} = 3\hat{i} + 2\hat{j} - 6\hat{k}$ $\vec{b} = 4\hat{i} - 3\hat{j} + \hat{k}$ and $\vec{c} = 4\hat{i} - 2\hat{j} + 4\hat{k}$

III. Write a program to find the angle between the following vectors:

1. $\vec{a} = \hat{i} + \hat{j}$ and $\vec{b} = \hat{j} + \hat{k}$ (Ans $\theta = \frac{\pi}{3}$)

2. $\vec{a} = 3\hat{i} + 2\hat{j} - 6\hat{k}$ and $\vec{b} = 4\hat{i} - 3\hat{j} + \hat{k}$ (Ans $\theta = \frac{\pi}{2}$)

3. $\vec{a} = 4\hat{i} - 2\hat{j} + 4\hat{k}$ and $\vec{b} = 3\hat{i} - 6\hat{j} - 2\hat{k}$ (Ans $\theta = \cos^{-1}(\frac{8}{21})$)

IV. Write a program to find the area of parallelogram whose sides are

1. $\vec{a} = \hat{i} - 2\hat{j} - 3\hat{k}$ $\vec{b} = 2\hat{i} + \hat{j} - \hat{k}$ (Ans $5\sqrt{3}$)

2. $\vec{a} = 3\hat{i} + 2\hat{j} - 6\hat{k}$ $\vec{b} = 4\hat{i} - 3\hat{j} + \hat{k}$ (Ans $7\sqrt{26}$)

V. Write a program to find the volume of parallelepiped whose edges are

1. $\vec{a} = \hat{i} - 2\hat{j} - 3\hat{k}$ $\vec{b} = 2\hat{i} + \hat{j} - \hat{k}$ and $\vec{c} = \hat{i} + 3\hat{j} - 2\hat{k}$ (Ans 20)

2. $\vec{a} = 3\hat{i} + 2\hat{j} - 6\hat{k}$ $\vec{b} = 4\hat{i} - 3\hat{j} + \hat{k}$ and $\vec{c} = 4\hat{i} - 2\hat{j} + 4\hat{k}$ (Ans 78)

Experiment 2

Program on Vector Differentiation and finding the Unit Tangent.

Aim: To find the derivatives of vector point functions and unit tangent vector using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
load ("vect")	vect is a package of functions for vector analysis. load ("vect") loads this package
<i>express (expr)</i>	Expands differential operator nouns into expressions in terms of partial derivatives. <i>express</i> recognizes the operators <i>grad</i> , <i>div</i> , <i>curl</i> , <i>laplacian</i> .
:=	To define a function/expression
<i>diff</i>	When <i>diff</i> is present as an <i>evflag</i> in call to <i>ev</i> , all differentiations indicated in <i>expr</i> are carried out.
~	The wedge product operator is denoted by the tilde ~. This is used to compute cross product of vectors.
trigsimp (<i>expr</i>)	Employs Pythagorean identities of trigonometric functions to simplify expressions.
* (asterik)	Commutative Multiplication
. (dot)	Noncommutative multiplication and scalar product
coeff (<i>expr</i> , <i>x</i>)	Returns the coefficient of <i>x</i> in <i>expr</i> , where <i>expr</i> is a polynomial or a monomial term in <i>x</i> .
apply('matrix, nested lists)	Converts nested lists of same length into a matrix
[<i>a_1</i> , ..., <i>a_n</i>];	To create a list [<i>a_1</i> , ..., <i>a_n</i>]
A[i]	[and] also enclose the subscripts of a list. A[i] will be i-th element of list A
diff (<i>expr</i> , <i>x</i>)	Returns the first partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
^ (Carrot Symbol) or **	For index/power/exponentiation (Commutative)
trigreduce (<i>expr</i> , <i>x</i>)	Combines products and powers of trigonometric and hyperbolic sin's and cos's of <i>x</i> into those of multiples of <i>x</i> .
ev (<i>expr</i> , <i>arg_1</i> , ..., <i>arg_n</i>)	Evaluates the expression <i>expr</i> in the environment specified by the arguments <i>arg_1</i> , ..., <i>arg_n</i> .
print ("text", <i>expr</i>)\$	Displays text within inverted commas and evaluates and displays <i>expr</i>
radcan(<i>expr</i>)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals
factorout (<i>expr</i> , <i>t</i>)	Rearranges the sum <i>expr</i> into a sum of terms of the form <i>f(t)*g</i> where <i>g</i> is a product of expressions not containing <i>t</i>
expand (<i>expr</i>)	Expand expression <i>expr</i> .

Definitions and Formulae:

Vector Function of a scalar variable: Let t be a scalar variable. If for each value of t there corresponds a unique vector $\vec{F}$, then $\vec{F}$ is called a vector function of the scalar variable t . If $\vec{F}$ is a space vector, then it will be of the form $\vec{F}(t) = f_1(t)\hat{i} + f_2(t)\hat{j} + f_3(t)\hat{k}$ where $f_1(t)$, $f_2(t)$ and $f_3(t)$ are functions of t .

Derivative of a Vector Function: Let $\vec{F}(t) = f_1(t)\hat{i} + f_2(t)\hat{j} + f_3(t)\hat{k}$ be a vector function of a scalar variable t . Then the derivatives of $\vec{F}(t)$ w.r.t. t are defined as:

$$\frac{d}{dt}(\vec{F}(t)) = \frac{d}{dt}(f_1(t))\hat{i} + \frac{d}{dt}(f_2(t))\hat{j} + \frac{d}{dt}(f_3(t))\hat{k}$$
$$\frac{d^2}{dt^2}(\vec{F}(t)) = \frac{d^2}{dt^2}(f_1(t))\hat{i} + \frac{d^2}{dt^2}(f_2(t))\hat{j} + \frac{d^2}{dt^2}(f_3(t))\hat{k}$$

and so on. Further, if t is time then $\frac{d}{dt}(\vec{F}(t))$ is the velocity and $\frac{d^2}{dt^2}(\vec{F}(t))$ is the acceleration at any time t .

Space Curve: Let $\vec{r}(t) = x(t)\hat{i} + y(t)\hat{j} + z(t)\hat{k}$ be the position vector of point $P(x(t), y(t), z(t))$. Then, $\vec{r}(t)$ represents a space curve.

Tangent Vector and the Unit Tangent Vector: Let $\vec{r}(t) = x(t)\hat{i} + y(t)\hat{j} + z(t)\hat{k}$ be a space curve. Then the derivative $\frac{d}{dt}(\vec{r}(t))$ at a point $P(x(t), y(t), z(t))$ represents a vector along the tangent to the curve at P . This is called tangent vector to $\vec{r}$ at P is denoted by $\vec{T}$. A unit vector in the direction of $\vec{T}$ is called the unit tangent vector at P . This unit tangent vector is denoted by $\hat{T}$. Thus,

$$\vec{T} = \frac{d}{dt}(\vec{r}(t)) = \frac{d}{dt}(x(t))\hat{i} + \frac{d}{dt}(y(t))\hat{j} + \frac{d}{dt}(z(t))\hat{k}$$
$$\hat{T} = \frac{\vec{T}}{|\vec{T}|}$$

Program:

Program to find the unit tangent vector to space curve $\vec{r} = f_1(t)\hat{i} + f_2(t)\hat{j} + f_3(t)\hat{k}$ at $t = t_0$.

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[f1(t), f2(t), f3(t)]$
r1:diff(r,t)$
T:r1.J/norm(r1)$
print("Space Curve r=", r.J)$
print("Unit Tangent Vector is T=",T)$
print("Unit Tangent Vector at given point is T=",expand(ev(T,t=t0)))$
```

Program to find the velocity and acceleration of space curve $\vec{r} = f_1(t)\hat{i} + f_2(t)\hat{j} + f_3(t)\hat{k}$ at $t = t_0$.

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[f1(t), f2(t), f3(t)]$
r1:diff(r,t)$
r2:diff(r1,t)$
print("Space Curve r=", r.J)$
print("Velocity is v=",r1.J)$
print("Acceleration is a=",r2.J)$
print("Magnitude of v at given point =", ev(norm(r1),t= t0))$
print("Magnitude of a at given point =", ev(norm(r2),t= t0))$
```

Worked Examples:

Problem 1. Write a program to find the unit tangent to the curve $\vec{r} = t^2\hat{i} + 2t\hat{j} - t^3\hat{k}$ at $t = 1$.

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[t^2,2*t,-t^3]$
r1:diff(r,t)$
T:r1.J/norm(r1)$
print("Space Curve r=", r.J)$
print("Unit Tangent Vector is T=",T)$
print("Unit Tangent Vector at given point is T=",expand(ev(T,t=1)))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[t^2,2*t,-t^3]$
r1:diff(r,t)$
T:r1.J/norm(r1)$
print("Space Curve r=", r.J)$
print("Unit Tangent Vector is T=",T)$
print("Unit Tangent Vector at given point is T=",expand(ev(T,t=1)))$

Space Curve r =  $-kt^3 + it^2 + 2jt$ 

Unit Tangent Vector is  $T = \frac{-3kt^2 + 2it + 2j}{\sqrt{9t^4 + 4t^2 + 4}}$ 

Unit Tangent Vector at given point is  $T = -\frac{3k}{\sqrt{17}} + \frac{2j}{\sqrt{17}} + \frac{2i}{\sqrt{17}}$ 
```

Output:

Space Curve $r = -kt^3 + it^2 + 2jt$

Unit Tangent Vector is $T = \frac{-3kt^2 + 2it + 2j}{\sqrt{9t^4 + 4t^2 + 4}}$

Unit Tangent Vector at given point is $T = -\frac{3k}{\sqrt{17}} + \frac{2j}{\sqrt{17}} + \frac{2i}{\sqrt{17}}$

Problem 2. Write a program to find the unit tangent to the curve $\vec{r} = \cos(3t)\hat{i} + \sin(3t)\hat{j} + 4t\hat{k}$ at $t = \frac{\pi}{4}$.

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[cos(3*t),sin(3*t),4*t]$
r1:diff(r,t)$
T:r1.J/norm(r1)$
print("Space Curve r=", r.J)$
print("Unit Tangent Vector is T=",expand(trigsimp(T)))$
print("Unit Tangent Vector at given point is T=",expand(ev(T,t=pi/4)))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[cos(3*t),sin(3*t),4*t]$
r1:diff(r,t)$
T:r1.J/norm(r1)$
print("Space Curve r=", r.J)$
print("Unit Tangent Vector is T=",expand(trigsimp(T)))$
print("Unit Tangent Vector at given point is T=",expand(ev(T,t=pi/4)))$

Space Curve r= j sin(3 t)+i cos(3 t)+4 k t

Unit Tangent Vector is T= -  $\frac{3 i \sin(3 t)}{5} + \frac{3 j \cos(3 t)}{5} + \frac{4 k}{5}$ 

Unit Tangent Vector at given point is T=  $\frac{4 k}{5} - \frac{3 j}{5 \sqrt{2}} - \frac{3 i}{5 \sqrt{2}}$ 
```

Output:

Space Curve $r = j \sin(3t) + i \cos(3t) + 4kt$

Unit Tangent Vector is $T = -\frac{3i \sin(3t)}{5} + \frac{3j \cos(3t)}{5} + \frac{4k}{5}$

Unit Tangent Vector at given point is $T = \frac{4k}{5} - \frac{3j}{5\sqrt{2}} - \frac{3i}{5\sqrt{2}}$

Problem 3. Write a program to find the velocity and acceleration of a particle moving along the curve

$\vec{r} = (1 - t^3)\hat{i} + (1 + t^2)\hat{j} + (2t - 5)\hat{k}$ at any time t .

Program:

```
load("vect")$
J:[i,j,k]$
r:[1-t^3, 1+t^2, 2*t-5]$
r1:diff(r,t)$
r2:diff(r1,t)$
print("Space Curve r=", r.J)$
print("Velocity is v=",r1.J)$
print("Acceleration is a=",r2.J)$
```

```
load("vect")$
J:[i,j,k]$
r:[1-t^3, 1+t^2, 2*t-5]$
r1:diff(r,t)$
r2:diff(r1,t)$
print("Space Curve r=", r.J)$
print("Velocity is v=",r1.J)$
print("Acceleration is a=",r2.J)$

vect: warning: removing existing rule or rules for ".".

Space Curve r= i (1-t^3)+j (t^2+1)+k (2t-5)

Velocity is v= -3 i t^2+2 j t+2 k

Acceleration is a= 2 j -6 i t
```

Output:

Space Curve $r = i(1 - t^3) + j(t^2 + 1) + k(2t - 5)$

Velocity is $v = -3it^2 + 2jt + 2k$

Acceleration is $a = 2j - 6it$

Problem 4. Write a program to find the velocity and acceleration of a particle moving along the curve

$\vec{r} = e^{-t} \hat{i} + 2\cos(3t)\hat{j} + 2\sin(3t)\hat{k}$ at any time t . Also find the magnitude of velocity and acceleration at $t = 0$

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[exp(-t), 2*cos(3*t), 2*sin(3*t)]$
r1:diff(r,t)$
r2:diff(r1,t)$
print("Space Curve r=", r.J)$
print("Velocity is v=",r1.J)$
print("Acceleration is a=",r2.J)$
print("Velocity at given point is v=",ev(r1.J, t=0))$
print("Acceleration at given point is a=",ev(r2.J,t=0))$
print("Magnitude of v at given point =", ev(norm(r1),t=0))$
print("Magnitude of a at given point =", ev(norm(r2),t=0))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[exp(-t), 2*cos(3*t), 2*sin(3*t)]$
r1:diff(r,t)$
r2:diff(r1,t)$
print("Space Curve r=", r.J)$
print("Velocity is v=",r1.J)$
print("Acceleration is a=",r2.J)$
print("Velocity at given point is v=",ev(r1.J, t=0))$
print("Acceleration at given point is a=",ev(r2.J,t=0))$
print("Magnitude of v at given point =", ev(norm(r1),t=0))$
print("Magnitude of a at given point =", ev(norm(r2),t=0))$
vect: warning: removing existing rule or rules for ".".
Space Curve r= 2 k sin(3 t)+2 j cos(3 t)+i %e-t
Velocity is v= -6 j sin(3 t)+6 k cos(3 t)-i %e-t
Acceleration is a= -18 k sin(3 t)-18 j cos(3 t)+i %e-t
Velocity at given point is v= 6 k -i
Acceleration at given point is a= i -18 j
Magnitude of v at given point =  $\sqrt{37}$ 
Magnitude of a at given point =  $5\sqrt{13}$ 
```

Output:

Space Curve $r = 2k \sin(3t) + 2j \cos(3t) + i\%e^{-t}$
Velocity is $v = -6j \sin(3t) + 6k \cos(3t) - i\%e^{-t}$
Acceleration is $a = -18k \sin(3t) - 18j \cos(3t) + i\%e^{-t}$
Velocity at given point is $v = 6k - i$
Acceleration at given point is $a = i - 18j$
Magnitude of v at given point = $\sqrt{37}$
Magnitude of a at given point = $5\sqrt{13}$

Problem 5. Write a program to find the velocity and acceleration and their magnitudes of a particle moving along the curve $\vec{r} = (\sin t) \hat{i} + (\cos t) \hat{j} + t \hat{k}$ at any time t .

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[sin(t), cos(t), t]$
r1:diff(r,t)$
r2:diff(r1,t)$
print("Space Curve r=", r.J)$
print("Velocity is v=",r1.J)$
print("Acceleration is a=",r2.J)$
print("Magnitude of v at any point =", trigsimp(norm(r1)))$
print("Magnitude of a at any point =", trigsimp(norm(r2)))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
r:[sin(t), cos(t), t]$
r1:diff(r,t)$
r2:diff(r1,t)$
print("Space Curve r=", r.J)$
print("Velocity is v=",r1.J)$
print("Acceleration is a=",r2.J)$
print("Magnitude of v at any point =", trigsimp(norm(r1)))$
print("Magnitude of a at any point =", trigsimp(norm(r2)))$
vect: warning: removing existing rule or rules for ".":
Space Curve r= i sin(t)+j cos(t)+k t
Velocity is v= -j sin(t)+i cos(t)+k
Acceleration is a= -i sin(t)-j cos(t)
Magnitude of v at any point =  $\sqrt{2}$ 
Magnitude of a at any point = 1
```

Output:

Space Curve $r = i \sin(t) + j \cos(t) + kt$
 Velocity is $v = -j \sin(t) + i \cos(t) + k$
 Acceleration is $a = -i \sin(t) - j \cos(t)$
 Magnitude of v at any point $= \sqrt{2}$
 Magnitude of a at any point $= 1$

Exercise:

I. Write a program to find the unit tangent to the curve $\vec{r}$ at given point t :

1. $\vec{r} = (t^2 + 1) \hat{i} + (4t - 3) \hat{j} + (2t^2 - 6t) \hat{k}$ at $t = 2$.
2. $\vec{r} = 2t^2 \hat{i} + (t^2 - 4t) \hat{j} + (3t - 5) \hat{k}$ at $t = -1$.
3. $\vec{r} = t \hat{i} + t^2 \hat{j} + t^3 \hat{k}$ at $t = -1$.

II. Write a program to find the velocity and acceleration of a particle moving along the curve $\vec{r}$ at any time t . Also find the magnitude of velocity and acceleration at given t :

1. $\vec{r} = (2 \sin 3t) \hat{i} + (2 \cos 3t) \hat{j} + (8t) \hat{k}$ at any time t
2. $\vec{r} = (4 \cos t) \hat{i} + (4 \sin t) \hat{j} + (6t) \hat{k}$ at $t = 0$.
3. $\vec{r} = (t^3 - 4t) \hat{i} + (t^2 + 4t) \hat{j} + (8t^2 - 3t^3) \hat{k}$ at $t = 2$.

Experiment 3

Program to find Curvature and Torsion of a space curve.

Aim: To find the Curvature and Torsion for a space curve at a given point using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
load ("vect")	vect is a package of functions for vector analysis. load ("vect") loads this package
<i>express (expr)</i>	Expands differential operator nouns into expressions in terms of partial derivatives. <i>express</i> recognizes the operators <i>grad</i> , <i>div</i> , <i>curl</i> , <i>laplacian</i> .
:=	To define a function/expression
<i>diff</i>	When <i>diff</i> is present as an <i>evflag</i> in call to <i>ev</i> , all differentiations indicated in <i>expr</i> are carried out.
~	The wedge product operator is denoted by the tilde ~. This is used to compute cross product of vectors.
trigsimp (<i>expr</i>)	Employs Pythagorean identities of trigonometric functions to simplify expressions.
* (asterik)	Commutative Multiplication
. (dot)	Noncommutative multiplication and scalar product
coeff (<i>expr</i> , <i>x</i>)	Returns the coefficient of <i>x</i> in <i>expr</i> , where <i>expr</i> is a polynomial or a monomial term in <i>x</i> .
apply('matrix, nested lists)	Converts nested lists of same length into a matrix
[<i>a_1</i> , ..., <i>a_n</i>];	To create a list [<i>a_1</i> , ..., <i>a_n</i>]
A[i]	[and] also enclose the subscripts of a list. A[i] will be i-th element of list A
diff (<i>expr</i> , <i>x</i>)	Returns the first partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
^ (Carrot Symbol) or **	For index/power/exponentiation (Commutative)
trigreduce (<i>expr</i> , <i>x</i>)	Combines products and powers of trigonometric and hyperbolic sin's and cos's of <i>x</i> into those of multiples of <i>x</i> . It also tries to eliminate these functions when they occur in denominators.
print ("text", <i>expr</i>)\$	Displays text within inverted commas and evaluates and displays <i>expr</i>
radcan(<i>expr</i>)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals
/ (Backward Slash)	Division
ev (<i>expr</i> , <i>arg_1</i> , ..., <i>arg_n</i>)	Evaluates the expression <i>expr</i> in the environment specified by the arguments <i>arg_1</i> , ..., <i>arg_n</i> .

Definitions and Formulae:

Fundamental vectors of a Space Curve: Let $\vec{r}(t) = x(t)\hat{i} + y(t)\hat{j} + z(t)\hat{k}$ be a space curve. With usual notations, let **T**, **N**, and **B** represent the **unit tangent**, the **principal normal** and the **binormal** respectively, to $\vec{r}(t)$ at any point t . These three unit vectors form a localized right-handed ($\mathbf{B}=\mathbf{T}\times\mathbf{N}$) coordinate system (TNB frame or Frenet Frame) at any specified point of $\vec{r}(t)$.

Curvature and Radius of Curvature of a space curve: For a space curve Let $\vec{r}(t) = x(t)\hat{i} + y(t)\hat{j} + z(t)\hat{k}$ the **unit tangent** **T** and the **principal normal** **N** are connected by the relation $\frac{dT}{ds} = \kappa \mathbf{N}$, where s is arc length parameter. Here κ is called the curvature and the quantity $\rho = \frac{1}{\kappa}$ is called the radius of curvature. Thus, $\kappa = \left| \frac{dT}{ds} \right| = \frac{|\vec{r}''(t) \times \vec{r}'''(t)|}{|\vec{r}'(t)|^3}$.

Torsion and Radius of Torsion of a space curve: For a space curve $\vec{r}(t) = x(t)\hat{i} + y(t)\hat{j} + z(t)\hat{k}$ the **binormal** **B** and the **principal normal** **N** are connected by the relation $\frac{dB}{ds} = -r\mathbf{N}$, where s is arc length parameter. Here r is called the torsion and the quantity $\sigma = \frac{1}{r}$ is called the radius of torsion.

$$\text{Thus, } r = \left| \frac{dB}{ds} \right| = \frac{[\vec{r}'(t), \vec{r}''(t), \vec{r}'''(t)]}{|\vec{r}'(t) \times \vec{r}''(t)|^2}.$$

Note: Greek Letters κ and r denote Curvature and Torsion respectively. But to avoid confusion between κ & $\hat{k}$ and r & t , in the following programs K is used for Curvature and T is used for Torsion.

Program:

Program to find the curvature and the torsion of $\vec{r}(t) = f_1(t)\hat{i} + f_2(t)\hat{j} + f_3(t)\hat{k}$ at given point $t = t_0$.

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[f1(t), f2(t), f3(t)]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K:norm(cross(r1,r2))/norm(r1)^3$
T:stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", radcan(K))$
print("Torsion is T=", radcan(T))$
print("Curvature at given point is K=", ev(K, t=t0))$
print("Torsion at given point is T=", ev(T, t=t0))$
```

Worked Examples:

Problem 1. Write a program to find the curvature and the torsion of the space curve $\vec{r} = t \hat{i} + t^2 \hat{j} + \frac{2}{3} t^3 \hat{k}$ at any point t and at $t = 0$.

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[t, t^2, 2/3*t^3]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K:norm(cross(r1,r2))/norm(r1)^3$
T:stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", radcan(K))$
print("Torsion is T=", radcan(T))$
print("Curvature at given point is K=", ev(K, t=0))$
print("Torsion at given point is T=", ev(T, t=0))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[t, t^2, 2/3*t^3]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K:norm(cross(r1,r2))/norm(r1)^3$
T:stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", radcan(K))$
print("Torsion is T=", radcan(T))$
print("Curvature at given point is K=", ev(K, t=0))$
print("Torsion at given point is T=", ev(T, t=0))$
```

$$\text{Space curve is } r = \frac{2kt^3}{3} + jt^2 + it$$

$$\text{Curvature is } K = \frac{2}{4t^4 + 4t^2 + 1}$$

$$\text{Torsion is } T = \frac{2}{4t^4 + 4t^2 + 1}$$

$$\text{Curvature at given point is } K = 2$$

$$\text{Torsion at given point is } T = 2$$

Output:

$$\text{Space curve is } r = \frac{2kt^3}{3} + jt^2 + it$$

$$\text{Curvature is } K = \frac{2}{4t^4 + 4t^2 + 1}$$

$$\text{Torsion is } T = \frac{2}{4t^4 + 4t^2 + 1}$$

$$\text{Curvature at given point is } K = 2$$

$$\text{Torsion at given point is } T = 2$$

Problem 2. Write a program to find the curvature and the torsion of the space curve $\vec{r} = t^2 \hat{i} + t^3 \hat{j} + t \hat{k}$ at any point t . Also, find the curvature and the torsion at $t=0$.

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
```

```

stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[t^2, t^3, t]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K:norm(cross(r1,r2))/norm(r1)^3$
T:stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", radcan(K))$
print("Torsion is T=", radcan(T))$
print("Curvature at given point is K=", ev(K, t=0))$
print("Torsion at given point is T=", ev(T, t=0))$

```

```

load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[t^2, t^3, t]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K:norm(cross(r1,r2))/norm(r1)^3$
T:stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", radcan(K))$
print("Torsion is T=", radcan(T))$
print("Curvature at given point is K=", ev(K, t=0))$
print("Torsion at given point is T=", ev(T, t=0))$

Space curve is r= j t^3 + i t^2 + k t

Curvature is K=  $\frac{2\sqrt{9t^4+9t^2+1}}{(9t^4+4t^2+1)^{3/2}}$ 

Torsion is T=  $\frac{3}{9t^4+9t^2+1}$ 
Curvature at given point is K= 2
Torsion at given point is T= 3

```

Output:

Space curve is $r = jt^3 + it^2 + kt$

Curvature is $K = \frac{2\sqrt{9t^4+9t^2+1}}{(9t^4+4t^2+1)^{3/2}}$

Torsion is $T = \frac{3}{9t^4+9t^2+1}$

Curvature at given point is $K = 2$

Torsion at given point is $T = 3$

Problem 3. Write a program to find the curvature and the torsion of the space curve $\vec{r} = (t - \sin(t)) \hat{i} + (1 - \cos(t)) \hat{j} + 4 \sin\left(\frac{t}{2}\right) \hat{k}$ at any point t and at $t = \frac{\pi}{3}$

Program:

```

load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[t-sin(t), 1-cos(t), 4*sin(t/2)]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$

```

```

K: norm(cross(r1,r2))/norm(r1)^3$
T: stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", trigreduce(trigsimp(K)))$
print("Torsion is T=", trigrat(T))$
print("Curvature at given point is K=", ev(K, t=pi/3))$
print("Torsion at given point is T=", ev(T, t=pi/3))$

```

Output:

Space curve is $r = i(t - \sin(t)) + j(1 - \cos(t)) + 4k \sin\left(\frac{t}{2}\right)$

Curvature is $K = \frac{\sqrt{6 - 2 \cos(t)}}{8}$

Torsion is $T = -\frac{\cos\left(\frac{3t}{2}\right) - 9 \cos\left(\frac{t}{2}\right)}{8 \cos(t) - 24}$

Curvature at given point is $K = \frac{\sqrt{5}}{8}$

Torsion at given point is $T = -\frac{\frac{5}{32}}{40}$

```

) load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[t-sin(t), 1-cos(t), 4*sin(t/2)]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K: norm(cross(r1,r2))/norm(r1)^3$
T: stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", trigreduce(trigsimp(K)))$
print("Torsion is T=", trigrat(T))$
print("Curvature at given point is K=", ev(K, t=pi/3))$
print("Torsion at given point is T=", ev(T, t=pi/3))$
vect: warning: removing existing rule or rules for ".
Space curve is r= i (t-sin(t))+j (1-cos(t))+4 k sin(t/2)

Curvature is K= sqrt(6-2 cos(t))/8

Torsion is T= - (cos(3t/2)-9 cos(t/2))/(8 cos(t)-24)

Curvature at given point is K= sqrt(5)/8

Torsion at given point is T= - 5/32 / 40

```

Problem 4. Write a program to find the curvature and the torsion of the space curve

$\vec{r} = a \cos(t) \hat{i} + a \sin(t) \hat{j} + bt \hat{k}$ at any point t .

Program:

```

load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[a*cos(t), a*sin(t), b*t]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K: norm(cross(r1,r2))/norm(r1)^3$
T: stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$

```

```

load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[a*cos(t), a*sin(t), b*t]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K: norm(cross(r1,r2))/norm(r1)^3$
T: stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", radcan(trigsimp(K)))$
print("Torsion is T=", trigsimp(T))$
vect: warning: removing existing rule or rules for ".
Space curve is r= a j sin(t)+a i cos(t)+b k t

Curvature is K= a/(b^2+a^2)

Torsion is T= b/(b^2+a^2)

```

```
print("Curvature is K=", radcan(trigsimp(K)))$
print("Torsion is T=", trigsimp(T))$
```

Output:

Space curve is $r = aj \sin(t) + ai \cos(t) + bkt$

$$\text{Curvature is } K = \frac{a}{b^2 + a^2}$$

$$\text{Torsion is } T = \frac{b}{b^2 + a^2}$$

Problem 5. Write a program to find the curvature and the torsion of the space curve

$\vec{r} = 3 \cos(t) \hat{i} + 3 \sin(t) \hat{j} + 4t \hat{k}$ at any point t .

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[3*cos(t), 3*sin(t), 4*t]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K:norm(cross(r1,r2))/norm(r1)^3$
T:stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", trigsimp(K))$
print("Torsion is T=", trigsimp(T))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
cross(x,y):=ev(express(x~y),diff)$
stp(x,y,z):=cross(x,y).z$
J:[i,j,k]$
r:[3*cos(t), 3*sin(t), 4*t]$
r1:diff(r,t)$
r2:diff(r1,t)$
r3:diff(r2,t)$
K:norm(cross(r1,r2))/norm(r1)^3$
T:stp(r1,r2,r3)/norm(cross(r1,r2))^2$
print("Space curve is r=", r.J)$
print("Curvature is K=", trigsimp(K))$
print("Torsion is T=", trigsimp(T))$

vect: warning: removing existing rule or rules for ".".
Space curve is r= 3 j sin(t)+3 i cos(t)+4 k t

Curvature is K= 3/25

Torsion is T= 4/25
```

Output:

Given curve is $r = 3j \sin(t) + 3i \cos(t) + 4kt$

$$\text{Curvature is } K = \frac{3}{25}$$

$$\text{Torsion is } T = \frac{4}{25}$$

Exercise:

Write a program to find the curvature and the torsion of a space curve $\vec{r}$ at the given point t :

1. $\vec{r} = at \hat{i} + bt^2 \hat{j} + ct^3 \hat{k}$ at any point t

2. $\vec{r} = (t - \frac{t^3}{3}) \hat{i} + t^2 \hat{j} + (\frac{t^3}{3} + t) \hat{k}$ at any point t

3. $\vec{r} = \cos t \hat{i} + \sin t \hat{j} + t^2 \hat{k}$ at any point t

4. $\vec{r} = a \cos t \hat{i} + a \sin t \hat{j} + a \cot(\alpha) t \hat{k}$ at any point t

5. $\vec{r} = a \cos(t) \hat{i} + a \sin(t) \hat{j}$ at any point t

6. $\vec{r} = (2t + 3) \hat{i} + (5 - t^2) \hat{j}$ at any point t

Experiment 4

Program to find the Gradient and the Laplacian of a scalar function and Divergence and Curl of a vector function.

Aim: To find the Gradient and the Laplacian of a scalar function and Divergence and Curl of a vector function using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
load ("vect")	vect is a package of functions for vector analysis. load ("vect") loads this package
<i>express (expr)</i>	Expands differential operator nouns into expressions in terms of partial derivatives. <i>express</i> recognizes the operators <i>grad</i> , <i>div</i> , <i>curl</i> , <i>laplacian</i> .
:=	To define a function/expression
<i>diff</i>	When <i>diff</i> is present as an <i>evflag</i> in call to <i>ev</i> , all differentiations indicated in <i>expr</i> are carried out.
grad()	gradient operator
div()	divergence operator
laplacian()	Laplacian operator
curl()	curl operator
~	The wedge product operator is denoted by the tilde ~. This is used to compute cross product of vectors.
trigsimp (<i>expr</i>)	Employs Pythagorean identities of trigonometric functions to simplify expressions.
trigreduce (<i>expr</i> , x)	Combines products and powers of trigonometric and hyperbolic sin's and cos's of x into those of multiples of x.
[a_1,...,a_n];	To create a list [a_1,...,a_n]
A[i]	[and] also enclose the subscripts of a list. A[i] will be i-th element of list A
sqrt()	square root of argument
print ("text", <i>expr</i>)\$	Displays text within inverted commas and evaluates and displays <i>expr</i>
radcan(<i>expr</i>)	Simplifies <i>expr</i> , which can contain logs, exponentials, and radicals
/ (Backward Slash)	Division
<i>ev (expr, arg_1, ..., arg_n)</i>	Evaluates the expression <i>expr</i> in the environment specified by the arguments <i>arg_1</i> , ..., <i>arg_n</i> .

Definitions and Formulae:

Scalar Point Function/Scalar Field: If to each point (x, y, z) of a region R in space there corresponds a number or scalar $\phi(x, y, z)$, then ϕ is called a scalar function of position or a scalar point function we say that a scalar field ϕ has been defined in R .

Vector Point Function/ Vector Field: If to each point (x, y, z) of a region R in space there corresponds a vector $\mathbf{f}(x, y, z)$, then $\mathbf{f}$ is called a vector function of position or a vector point function we say that a scalar field $\mathbf{f}$ has been defined in R .

Gradient of a scalar field: Let $\phi(x, y, z)$ be defined and differentiable at each point (x, y, z) in a certain region of space then the *gradient* of ϕ , written $\nabla\phi$ or *grad* ϕ , is a vector field defined by

$$\nabla\phi = \left(\frac{\partial}{\partial x} \hat{i} + \frac{\partial}{\partial y} \hat{j} + \frac{\partial}{\partial z} \hat{k}\right)\phi = \frac{\partial\phi}{\partial x} \hat{i} + \frac{\partial\phi}{\partial y} \hat{j} + \frac{\partial\phi}{\partial z} \hat{k}$$

Laplacian of a scalar field: Let $\phi(x, y, z)$ be defined and differentiable at each point (x, y, z) in a certain region of space then the *Laplacian* of ϕ , written $\nabla^2\phi$ is a scalar field defined by

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}$$

Divergent of a vector field: Let $\vec{f} = f_1 \hat{i} + f_2 \hat{j} + f_3 \hat{k}$ be defined and differentiable at each point (x, y, z) in a certain region of space then the *divergence* of $\vec{f}$, written $\nabla \cdot \vec{f}$ or *div* $\vec{f}$, is a scalar field defined by

$$\nabla \cdot \vec{f} = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} + \frac{\partial f_3}{\partial z}$$

Curl/ rotation of a vector field: Let $\vec{f} = f_1 \hat{i} + f_2 \hat{j} + f_3 \hat{k}$ be defined and differentiable at each point (x, y, z) in a certain region of space then the *curl or rotation* of $\vec{f}$, written $\nabla \times \vec{f}$ or *curl* $\vec{f}$ or *rot* $\vec{f}$ is a vector field defined by

$$\nabla \times \vec{f} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ f_1 & f_2 & f_3 \end{vmatrix}$$

Program:

Program to find the gradient and Laplacian of a scalar field Φ at a given point (x_0, y_0, z_0) .

```
load ("vect")$
J:[i,j,k]$
Φ:given scalar field$
G:ev(express(grad (Φ)),diff)$
L:ev(express(laplacian (Φ)),diff)$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("laplacian(Φ)=", L)$
print("at given point, grad(Φ)=", ev(G,x=x0,y=y0,z=z0).J)$
print("at given point, laplacian(Φ)=", ev(L, x=x0,y=y0,z=z0))$
```

Program to find the divergence and curl of a vector field $\vec{f} = f_1 \hat{i} + f_2 \hat{j} + f_3 \hat{k}$ at the given point (x_0, y_0, z_0) .

```
load ("vect")$
J:[i,j,k]$
f:[f1, f2, f3]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(Φ)=",C.J)$
print("at given point, div(Φ)=", ev(D, x=x0,y=y0,z=z0))$
print("at given point, curl(Φ)=", ev(C, x=x0,y=y0,z=z0).J)$
```

Worked Examples:

Problem 1. Write a program to find the gradient and Laplacian of $\Phi = 3x^2y - y^3z^2$ at $(1, -2, -1)$

Program:

```
load ("vect")$
J:[i,j,k]$
Φ:3*x^2*y-y^3*z^2$
G:ev(express(grad (Φ)),diff)$
L:ev(express(laplacian (Φ)),diff)$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("laplacian(Φ)=", L)$
print("at given point, grad(Φ)=", ev(G,x=1,y=-2,z=-1).J)$
print("at given point, laplacian(Φ)=", ev(L,x=1,y=-2,z=-1))$
```

```
load ("vect")$
J:[i,j,k]$
Φ:3*x^2*y-y^3*z^2$
G:ev(express(grad (Φ)),diff)$
L:ev(express(laplacian (Φ)),diff)$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("laplacian(Φ)=", L)$
print("at given point, grad(Φ)=", ev(G,x=1,y=-2,z=-1).J)$
print("at given point, laplacian(Φ)=", ev(L,x=1,y=-2,z=-1))$

Φ = 3 x^2 y - y^3 z^2
grad(Φ) = j (3 x^2 - 3 y^2 z^2) - 2 k y^3 z + 6 i x y
laplacian(Φ) = -6 y z^2 - 2 y^3 + 6 y
at given point, grad(Φ) = -16 k - 9 j - 12 i
at given point, laplacian(Φ) = 16
```

Output:

$\Phi = 3x^2y - y^3z^2$
 $grad(\Phi) = j(3x^2 - 3y^2z^2) - 2ky^3z + 6ixy$
 $laplacian(\Phi) = -6yz^2 - 2y^3 + 6y$
at given point, $grad(\Phi) = -16k - 9j - 12i$
at given point, $laplacian(\Phi) = 16$

Problem 2. Write a program to find the gradient and Laplacian of $\Phi = x^2yz$ at $(1, -2, 1)$

Program:

```
load ("vect")$
J:[i,j,k]$
Φ:x^2*y*z$
G:ev(express(grad (Φ)),diff)$
L:ev(express(laplacian (Φ)),diff)$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("laplacian(Φ)=", L)$
print("at given point, grad(Φ)=", ev(G,x=1,y=-2,z=1).J)$
print("at given point, laplacian(Φ)=", ev(L,x=1,y=-2,z=1))$
```

```
load ("vect")$
J:[i,j,k]$
Φ:x^2*y*z$
G:ev(express(grad (Φ)),diff)$
L:ev(express(laplacian (Φ)),diff)$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("laplacian(Φ)=", L)$
print("at given point, grad(Φ)=", ev(G,x=1,y=-2,z=1).J)$
print("at given point, laplacian(Φ)=", ev(L,x=1,y=-2,z=1))$
vect: warning: removing existing rule or rules for ". ".

Φ = x^2 y z
grad(Φ) = 2 i x y z + j x^2 z + k x^2 y
laplacian(Φ) = 2 y z
at given point, grad(Φ) = -2 k + j - 4 i
at given point, laplacian(Φ) = -4
```

Output:

$\Phi = x^2yz$

$$\text{grad}(\Phi) = 2ixyz + jx^2z + kx^2y$$

$$\text{laplacian}(\Phi) = 2yz$$

$$\text{at given point, grad}(\Phi) = -2k + j - 4i$$

$$\text{at given point, laplacian}(\Phi) = -4$$

Problem 3. Write a program to find the gradient and Laplacian of $\Phi = 2xy + 5yz + zx$ at $(1,2,3)$

Program:

```
load ("vect")$
J:[i,j,k]$
Phi:2*x*y+5*y*z+z*x$
G:ev(express(grad (Phi)),diff)$
L:ev(express(laplacian (Phi)),diff)$
print("Phi=", Phi)$
print("grad(Phi)=", G.J)$
print("laplacian(Phi)=", L)$
print("at given point, grad(Phi)=", ev(G,x=1,y=2,z=3).J)$
print("at given point, laplacian(Phi)=", ev(L,x=1,y=2,z=3))$
```

```
load ("vect")$
J:[i,j,k]$
Phi:2*x*y+5*y*z+z*x$
G:ev(express(grad (Phi)),diff)$
L:ev(express(laplacian (Phi)),diff)$
print("Phi=", Phi)$
print("grad(Phi)=", G.J)$
print("laplacian(Phi)=", L)$
print("at given point, grad(Phi)=", ev(G,x=1,y=2,z=3).J)$
print("at given point, laplacian(Phi)=", ev(L,x=1,y=2,z=3))$

vect: warning: removing existing rule or rules for ".".
Phi= 5 y z + x z + 2 x y
grad(Phi)= j (5 z + 2 x) + i (z + 2 y) + k (5 y + x)
laplacian(Phi)= 0
at given point, grad(Phi)= 11 k + 17 j + 7 i
at given point, laplacian(Phi)= 0
```

Output:

$$\Phi = 5yz + xz + 2xy$$

$$\text{grad}(\Phi) = j(5z + 2x) + i(z + 2y) + k(5y + x)$$

$$\text{laplacian}(\Phi) = 0$$

$$\text{at given point, grad}(\Phi) = 11k + 17j + 7i$$

$$\text{at given point, laplacian}(\Phi) = 0$$

Problem 4. Write a program to find divergence and curl of $\vec{f} = x^2z \hat{i} - 2y^3z^2 \hat{j} + xy^2z \hat{k}$ at $(1, -1, 1)$

Program:

```
load ("vect")$
J:[i,j,k]$
f:[x^2*z,-2*y^3*z^2, x*y^2*z]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=", C.J)$
```

```
load ("vect")$
J:[i,j,k]$
f:[x^2*z,-2*y^3*z^2, x*y^2*z]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=", C.J)$
print("at given point, div(f)=", ev(D,x=1,y=-1,z=1))$
print("at given point, curl(f)=", ev(C,x=1,y=-1,z=1).J)$

vect: warning: removing existing rule or rules for ".".
f= -2 j y^3 z^2 + k x y^2 z + i x^2 z
div(f)= -6 y^2 z^2 + 2 x z + x y^2
curl(f)= i (4 y^3 z + 2 x y z) + j (x^2 - y^2 z)
at given point, div(f)= -3
at given point, curl(f)= -6 i
```

```
print("at given point, div(f)=", ev(D,x=1,y=-1,z=1))$
print("at given point, curl(f)=", ev(C,x=1,y=-1,z=1).J)$
```

Output:

```
f = -2jy3z2 + kxy2z + ix2z
div(f) = -6y2z2 + 2xz + xy2
curl(f) = i(4y3z + 2xyz) + j(x2 - y2z)
at given point, div(f) = -3
at given point, curl(f) = -6i
```

Problem 5. Write a program to find divergence and curl of $\vec{f} = xz^3 \hat{i} - 2x^2yz \hat{j} + 2yz^4 \hat{k}$ at $(1, -1, 1)$

Program:

```
load ("vect")$
J:[i,j,k]$
f:[x*z^3, -2*x^2*y*z, 2*y*z^4]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=", C.J)$
print("at given point, div(f)=", ev(D,x=1,y=-1,z=1))$
print("at given point, curl(f)=", ev(C,x=1,y=-1,z=1).J)$
```

```
load ("vect")$
J:[i,j,k]$
f:[x*z^3, -2*x^2*y*z, 2*y*z^4]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=", C.J)$
print("at given point, div(f)=", ev(D,x=1,y=-1,z=1))$
print("at given point, curl(f)=", ev(C,x=1,y=-1,z=1).J)$

f= 2 k y z4 + i x z3 - 2 j x2 y z
div(f)= 8 y z3 + z3 - 2 x2 z
curl(f)= i (2 z4 + 2 x2 y) + 3 j x z2 - 4 k x y z
at given point, div(f)= -9
at given point, curl(f)= 4 k + 3 j
```

Output:

```
f = 2kyz4 + ixz3 - 2jx2yz
div(f) = 8yz3 + z3 - 2x2z
curl(f) = i(2z4 + 2x2y) + 3jxz2 - 4kxyz
at given point, div(f) = -9
at given point, curl(f) = 4k + 3j
```

Problem 6. Write a program to find divergence and curl of $\text{grad}(\Phi)$ where $\Phi = x^3 + y^3 - 3xyz$

Program:

```
load ("vect")$
J:[i,j,k]$
Phi:x^3+y^3+z^3-3*x*y*z$
G:ev(express(grad(Phi)),diff)$
D:ev(express(div(G)),diff)$
```

```

C:ev(express(curl(G)),diff)$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("div(grad(Φ))=", D)$
print("curl(grad(Φ))=", C.J)$

```

Output:

$$\Phi = z^3 - 3xyz + y^3 + x^3$$

$$\text{grad}(\Phi) = k(3z^2 - 3xy) + i(3x^2 - 3yz) + j(3y^2 - 3xz)$$

$$\text{div}(\text{grad}(\Phi)) = 6z + 6y + 6x$$

$$\text{curl}(\text{grad}(\Phi)) = 0$$

```

load ("vect")$
J:[i,j,k]$
Φ:x^3+y^3+z^3-3·x·y·z$
G:ev(express(grad (Φ)),diff)$
D:ev(express(div(G)),diff)$
C:ev(express(curl(G)),diff)$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("div(grad(Φ))=", D)$
print("curl(grad(Φ))=",C.J)$

Φ= z3-3 x y z+y3+x3
grad(Φ)= k (3 z2-3 x y)+i (3 x2-3 y z)+j (3 y2-3 x z)
div(grad(Φ))= 6 z+6 y+6 x
curl(grad(Φ))= 0

```

Problem 7. Write a program to find $\text{div}(\text{curl}(\vec{f}))$ and $\text{curl}(\text{curl}(\vec{f}))$ at (2,1,1) where

$$\vec{f} = x^2yz \hat{i} + xyz^2 \hat{j} + y^2z \hat{k}$$

Program:

```

load ("vect")$
J:[i,j,k]$
f:[x^2·y·z, x·y·z^2, y^2·z]$
C:ev(express(curl(f)),diff)$
D:ev(express(div(C)),diff)$
CC:ev(express(curl(C)),diff)$
print("f=", f.J)$
print("curl(f)=", C.J)$
print("div(curl(f))=", D)$
print("curl(curl(f))=", CC.J)$
print("at given point, div(curl(f))=", ev(D,x=2,y=1,z=1))$
print("at given point, curl(curl(f))=", ev(CC,x=2,y=1,z=1).J)$

```

```

load ("vect")$
J:[i,j,k]$
f:[x^2·y·z, x·y·z^2, y^2·z]$
C:ev(express(curl(f)),diff)$
D:ev(express(div(C)),diff)$
CC:ev(express(curl(C)),diff)$
print("f=", f.J)$
print("curl(f)=", C.J)$
print("div(curl(f))=", D)$
print("curl(curl(f))=", CC.J)$
print("at given point, div(curl(f))=", ev(D,x=2,y=1,z=1))$
print("at given point, curl(curl(f))=", ev(CC,x=2,y=1,z=1).J)$

f= j x y z2+k y2 z+i x2 y z
curl(f)= k (y z2-x2 z)+i (2 y z-2 x y z)+j x2 y
div(curl(f))= 0
curl(curl(f))= i z2+k (2 x z-2 z+2 x y)+j (2 x z-2 x y+2 y)
at given point, div(curl(f))= 0
at given point, curl(curl(f))= 6 k+2 j+i

```

Output:

$$f = jxyz^2 + ky^2z + ix^2yz$$

$$\text{curl}(f) = k(yz^2 - x^2z) + i(2yz - 2xyz) + jx^2y$$

$$\text{div}(\text{curl}(f)) = 0$$

$$\text{curl}(\text{curl}(f)) = iz^2 + k(2xz - 2z + 2xy) + j(2xz - 2xy + 2y)$$

$$\text{at given point, div}(\text{curl}(f)) = 0$$

$$\text{at given point, curl}(\text{curl}(f)) = 6k + 2j + i$$

Exercise:

I. Write a program to find the gradient and Laplacian of the following scalar fields Φ at the given point:

1. $\Phi = x^2y^2z^4$ at $(3, 1, -2)$
2. $\Phi = x^3 + y^3 + 3xyz$ at $(1, 2, -1)$
3. $\Phi = x^2yz + 4xz^2$ at $(1, -2, -1)$
4. $\Phi = xy^2 + yz^3$ at $(2, -1, 1)$

II. Write a program to find the divergence and curl of the following vector fields $\vec{f}$ at the given point:

1. $\vec{f} = (x + y + 1)\hat{i} + \hat{j} - (x + y)\hat{k}$ at $(2, 1, -1)$
2. $\vec{f} = x^2y\hat{i} + yz^3\hat{j} - zx^3\hat{k}$ at $(1, 1, 1)$
3. $\vec{f} = \text{grad}(\Phi)$ where $\Phi = 2x^3y^2z^4$ at $(-1, 2, -3)$
4. $\vec{f} = \text{grad}(\Phi)$ where $\Phi = x^4 - 6x^2y^2 + y^4$ at $(1, 2, 3)$

III. Write a program to find the $\text{div}(\text{curl}(\vec{f}))$ and $\text{curl}(\text{curl}(\vec{f}))$ of the following vector fields $\vec{f}$ at the given point:

1. $\vec{f} = 2x^2z\hat{i} - xy^2z\hat{j} + 3yz^2\hat{k}$ at $(1, 1, 2)$
2. $\vec{f} = x^2y\hat{i} + y^2z\hat{j} + z^2x\hat{k}$ at $(-2, 1, 1)$

Experiment 5

Program to demonstrate the physical interpretation of Gradient, Divergence and Curl.

Aim: To demonstrate physical interpretations of Gradient, Divergence and Curl like directional derivative, unit normal, solenoidal, irrotational using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
load ("vect")	vect is a package of functions for vector analysis. load ("vect") loads this package
<i>express (expr)</i>	Expands differential operator nouns into expressions in terms of partial derivatives. <i>express</i> recognizes the operators <i>grad</i> , <i>div</i> , <i>curl</i> , <i>laplacian</i> .
:=	To define a function/expression
<i>diff</i>	When <i>diff</i> is present as an <i>evflag</i> in call to <i>ev</i> , all differentiations indicated in <i>expr</i> are carried out.
grad()	gradient operator
div()	divergence operator
laplacian()	Laplacian operator
curl()	curl operator
~	The wedge product operator is denoted by the tilde ~. This is used to compute cross product of vectors.
trigsimp (<i>expr</i>)	Employs Pythagorean identities of trigonometric functions to simplify expressions.
trigreduce (<i>expr</i> , <i>x</i>)	Combines products and powers of trigonometric and hyperbolic sin's and cos's of <i>x</i> into those of multiples of <i>x</i> .
* (asterik)	Commutative Multiplication
. (dot)	Noncommutative multiplication and scalar product
coeff (<i>expr</i> , <i>x</i>)	Returns the coefficient of <i>x</i> in <i>expr</i> , where <i>expr</i> is a polynomial or a monomial term in <i>x</i> .
apply('matrix, nested lists)	Converts nested lists of same length into a matrix
[<i>a_1</i> , ..., <i>a_n</i>];	To create a list [<i>a_1</i> , ..., <i>a_n</i>]
A[<i>i</i>]	[and] also enclose the subscripts of a list. A[<i>i</i>] will be <i>i</i> -th element of list A
sqrt()	square root of argument
diff (<i>expr</i> , <i>x</i>)	Returns the first partial derivative of <i>expr</i> with respect to the variable <i>x</i> .
/ (Backward Slash)	Division
acos()	arc cos or $\cos^{-1}$ function

Definitions and Formulae:

Level Surfaces: If $\Phi(x, y, z)$ is a scalar point function, then $\Phi(x, y, z) = C$, where C is a constant is called a level surface of the function $\Phi(x, y, z)$. At every point on the level surface, the function $\Phi(x, y, z)$ takes a constant value C .

Physical Interpretation of Gradient:

a) Directional Derivative: The component of $\text{grad}(\Phi)$ of a scalar field Φ in the direction of $\vec{a}$ is given by $\text{grad}(\Phi) \cdot \hat{a}$ and is called the directional derivative of Φ in the direction of $\vec{a}$. Here $\hat{a}$ is the unit vector in the direction of $\vec{a}$. Physically, this is the rate of change of Φ at (x, y, z) in the direction of $\vec{a}$.

b) Unit normal: The $\text{grad}(\Phi)$ is a vector normal to the level surface $\Phi = C$ and $\hat{n} = \frac{\text{grad}(\Phi)}{|\text{grad}(\Phi)|}$ is the unit normal vector to the surface $\Phi = C$ at any point (x, y, z) .

c) Angle between surfaces: Angle between two surfaces is the angle between their normals at the common point. At the common point (x_0, y_0, z_0) , angle between $\Phi = C_1$ and $\psi = C_2$ is given by $\theta = \cos^{-1} \left(\frac{\text{grad}(\Phi) \cdot \text{grad}(\psi)}{|\text{grad}(\Phi)| \cdot |\text{grad}(\psi)|} \right)$ evaluated at (x_0, y_0, z_0) .

Physical Interpretation of Divergent: If $\vec{f}$ represents the velocity of a fluid then $\text{div}(\vec{f})$ represents the rate at which the fluid is decreasing per unit volume. A vector point function $\vec{f}$ is said to be solenoidal if $\text{div}(\vec{f}) = 0$.

Physical Interpretation of Curl: If $\vec{f}$ represents the velocity of a body rotating about a fixed point then $\text{curl}(\vec{f})$ represents twice the angular velocity. A vector point function $\vec{f}$ is said to be irrotational if $\text{curl}(\vec{f}) = 0$.

Program:

Program to find the directional derivative of Φ in the direction of $\vec{a} = a_1 \hat{i} + a_2 \hat{j} + a_3 \hat{k}$ at the given point (x_0, y_0, z_0) .

```
load ("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Phi:given scalar field$
a:[a1, a2, a3]$
G:ev(express(grad (Phi)),diff)$
DD:G.a/norm(a)$
print("Phi=", Phi)$
print("a=", a.J)$
print("grad(Phi)=", G.J)$
print("Directional Derivative of Phi=", radcan(DD))$
print("at the given point, Directional Derivative of Phi=", ev(DD,x=x0,y=y0,z=z0))$
```

Program to find the unit normal to the surface $\Phi = C$ at the given point (x_0, y_0, z_0) .

```
load ("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Φ:given scalar field (i.e., LHS of Φ=C)$
G:ev(express(grad (Φ)),diff)$
n:radcan(G/norm(G))$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("Unit Normal to Φ=", expand(n.J))$
print("at the given point, Unit Normal to Φ=", ev(n, x=x0,y=y0,z=z0).J)$
```

Program to find the angle between surfaces $\Phi = C_1$ and $\Psi = C_2$ at the common point (x_0, y_0, z_0) .

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
Φ: given scalar field (i.e., LHS of Φ=C1)$
Ψ: given scalar field (i.e., LHS of Ψ=C2)$
G1:ev(express(grad (Φ)),diff)$
G2:ev(express(grad (Ψ)),diff)$
θ:acos(G1.G2/(norm(G1)*norm(G2)))$
print("Φ=", Φ)$
print("Ψ=", Ψ)$
print("Angle between Φ and Ψ is θ=", ev(θ, x=x0,y=y0,z=z0))$
```

Program to test whether given vector field $\vec{f} = f_1 \hat{i} + f_2 \hat{j} + f_3 \hat{k}$ is solenoidal/irrotational or not.

```
load ("vect")$
J:[i,j,k]$
f:[f1,f2, f3]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=",C.J)$
```

```

if D=0 then print("Given Vector field is solenoidal")
else print("Given Vector field is not solenoidal")$
if C.J=0 then print("Given Vector field is irrotational")
else print("Given Vector field is not irrotational")$

```

Worked Examples:

Problem 1. Write a program to find the directional derivative of $\Phi = x^2yz + 4xz^2$ in the direction of $\vec{a} = 2\hat{i} - \hat{j} - 2\hat{k}$ at the given point $(1, -2, -1)$.

Program:

```

load ("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Φ:x^2*y*z+4*x*z^2$
a:[2,-1,-2]$
G:ev(express(grad (Φ)),diff)$
DD:G.a/norm(a)$
print("Φ=", Φ)$
print("a=", a.J)$
print("grad(Φ)=", G.J)$
print("Directional Derivative of Φ=", radcan(DD))$
print("at the given point, Directional Derivative of Φ=", ev(DD,x=1,y=-2,z=-1))$

```

```

load ("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Φ:x^2*y*z+4*x*z^2$
a:[2,-1,-2]$
G:ev(express(grad (Φ)),diff)$
DD:G.a/norm(a)$
print("Φ=", Φ)$
print("a=", a.J)$
print("grad(Φ)=", G.J)$
print("Directional Derivative of Φ=", radcan(DD))$
print("at the given point, Directional Derivative of Φ=", ev(DD,x=1,y=-2,z=-1))$
vect: warning: removing existing rule or rules for ".".

Φ= 4 x z^2 + x^2 y z
a= -2 k - j + 2 i
grad(Φ)= i (4 z^2 + 2 x y z) + k (8 x z + x^2 y) + j x^2 z

Directional Derivative of Φ=  $\frac{8 z^2 + (4 x y - x^2 - 16 x) z - 2 x^2 y}{3}$ 

at the given point, Directional Derivative of Φ=  $\frac{37}{3}$ 

```

Output:

```

Φ = 4xz2 + x2yz
a = -2k - j + 2i
grad(Φ) = i(4z2 + 2xyz) + k(8xz + x2y) + jx2z
Directional Derivative of Φ =  $\frac{8z^2 + (4xy - x^2 - 16x)z - 2x^2y}{3}$ 
at the given point, Directional Derivative of Φ =  $\frac{37}{3}$ 

```

Problem 2. Write a program to find the directional derivative of $\Phi = 2xy + 5yz + zx$ in the direction of $\vec{a} = 3\hat{i} - 5\hat{j} + 4\hat{k}$ at the given point $(1, 2, 3)$.

Program:

```
load ("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Φ:2*x*y+5*y*z+z*x$
a:[3,-5,4]$
G:ev(express(grad (Φ)),diff)$
DD:G.a/norm(a)$
print("Φ=", Φ)$
print("a=", a.J)$
print("grad(Φ)=", G.J)$
print("Directional Derivative of Φ=", radcan(DD))$
print("at the given point, Directional Derivative of Φ=", ev(DD,x=1,y=2,z=3))$
```

```
load ("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Φ:2*x*y+5*y*z+z*x$
a:[3,-5,4]$
G:ev(express(grad (Φ)),diff)$
DD:G.a/norm(a)$
print("Φ=", Φ)$
print("a=", a.J)$
print("grad(Φ)=", G.J)$
print("Directional Derivative of Φ=", radcan(DD))$
print("at the given point,Directional Derivative of Φ=", ev(DD,x=1,y=2,z=3))$
vect: warning: removing existing rule or rules for ".".
Φ= 5 y z + x z + 2 x y
a= 4 k - 5 j + 3 i
grad(Φ)= j (5 z + 2 x) + i (z + 2 y) + k (5 y + x)
Directional Derivative of Φ= -  $\frac{22 z - 26 y + 6 x}{5 \sqrt{2}}$ 
at the given point,Directional Derivative of Φ=  $-2^{\frac{3}{2}}$ 
```

Output:

$\Phi = 5yz + xz + 2xy$
 $a = 4k - 5j + 3i$
 $grad(\Phi) = j(5z + 2x) + i(z + 2y) + k(5y + x)$
 $Directional\ Derivative\ of\ \Phi = -\frac{22z - 26y + 6x}{5\sqrt{2}}$
 $at\ the\ given\ point, Directional\ Derivative\ of\ \Phi = -2^{\frac{3}{2}}$

Problem 3. Write a program to find the unit normal to the surface $xy^3z^2 = 4$ at $(-1, -1, 2)$.

Program:

```
load ("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Φ:x*y^3*z^2$
G:ev(express(grad (Φ)),diff)$
n:radcan(G/norm(G))$
print("Φ=", Φ)$
print("grad(Φ)=", G.J)$
print("Unit Normal of Φ=", expand(n.J))$
print("at the given point, Unit Normal of Φ=", ev(n,x=-1,y=-1,z=2).J)$
```

Output:

$$\Phi = xy^3z^2$$

$$\text{grad}(\Phi) = iy^3z^2 + 3jxy^2z^2 + 2kxy^3z$$

$$\text{Unit Normal to } \Phi = \frac{iyz}{\sqrt{y^2z^2 + 9x^2z^2 + 4x^2y^2}} + \frac{3jxz}{\sqrt{y^2z^2 + 9x^2z^2 + 4x^2y^2}} + \frac{2kxy}{\sqrt{y^2z^2 + 9x^2z^2 + 4x^2y^2}}$$

$$\text{at the given point, Unit Normal to } \Phi = \frac{k}{\sqrt{11}} - \frac{3j}{\sqrt{11}} - \frac{i}{\sqrt{11}}$$

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
J:[i,j,k]$
Phi:x*y^3*z^2$
G:ev(express(grad(Phi)),diff)$
n:radcan(G/norm(G))$
print("Phi=",Phi)$
print("grad(Phi)=",G.J)$
print("Unit Normal of Phi=",expand(n.J))$
print("at the given point,Unit Normal of Phi=",ev(n,x=-1,y=-1,z=2).J)$

Phi= x y^3 z^2
grad(Phi)= i y^3 z^2 + 3 j x y^2 z^2 + 2 k x y^3 z
Unit Normal of Phi= \frac{iyz}{\sqrt{y^2z^2+9x^2z^2+4x^2y^2}} + \frac{3jxz}{\sqrt{y^2z^2+9x^2z^2+4x^2y^2}} + \frac{2kxy}{\sqrt{y^2z^2+9x^2z^2+4x^2y^2}}
at the given point,Unit Normal of Phi= \frac{k}{\sqrt{11}} - \frac{3j}{\sqrt{11}} - \frac{i}{\sqrt{11}}
```

Problem 4. Write a program to find the angle between surfaces $x^2 - 2y^2 + 4z^2 = 3$ and $xyz^2 = 1$ at the common point $(1, 1, -1)$.

Program:

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
Phi:x^2-2*y^2+4*z^2$
Psi:x*y*z^2$
G1:ev(express(grad(Phi)),diff)$
G2:ev(express(grad(Psi)),diff)$
theta:acos(G1.G2/(norm(G1)*norm(G2)))$
print("Phi=",Phi)$
print("Psi=",Psi)$
print("Angle between Phi and Psi is theta=",ev(theta,x=1,y=1,z=-1))$
```

```
load("vect")$
norm(x):=sqrt(x[1]^2+x[2]^2+x[3]^2)$
Phi:x^2-2*y^2+4*z^2$
Psi:x*y*z^2$
G1:ev(express(grad(Phi)),diff)$
G2:ev(express(grad(Psi)),diff)$
theta:acos(G1.G2/(norm(G1)*norm(G2)))$
print("Phi=",Phi)$
print("Psi=",Psi)$
print("Angle between Phi and Psi is theta=",ev(theta,x=1,y=1,z=-1))$
vect: warning: removing existing rule or rules for "."
Phi= 4 z^2 - 2 y^2 + x^2
Psi= x y z^2
Angle between Phi and Psi is theta= acos\left(\frac{7}{\sqrt{6}\sqrt{21}}\right)
```

Output:

$$\Phi = 4z^2 - 2y^2 + x^2$$

$$\Psi = xyz^2$$

$$\text{Angle between } \Phi \text{ and } \Psi \text{ is } \theta = \cos\left(\frac{7}{\sqrt{6}\sqrt{21}}\right)$$

Problem 5. Write a program to test whether $\vec{f} = (x + 2y + 4z)\hat{i} + (2x - 3y - z)\hat{j} + (4x - y + 2z)\hat{k}$ is solenoidal/irrotational or not

Program:

```
load ("vect")$
J:[i,j,k]$
f:[x+2*y+4*z,2*x-3*y-z, 4*x-y+2*z]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=",C.J)$
if D=0 then print("Given Vector field is solenoidal")
else print("Given Vector field is not solenoidal")$
if C.J=0 then print("Given Vector field is irrotational")
else print("Given Vector field is not irrotational")$
```

```
load ("vect")$
J:[i,j,k]$
f:[x+2*y+4*z,2*x-3*y-z, 4*x-y+2*z]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=",C.J)$
if D=0 then print("Given Vector field is solenoidal")
else print("Given Vector field is not solenoidal")$
if C.J=0 then print("Given Vector field is irrotational")
else print("Given Vector field is not irrotational")$
vect: warning: removing existing rule or rules for ".
f= i (4 z+2 y+x)+k (2 z-y+4 x)+j (-z-3 y+2 x)
div(f)= 0
curl(f)= 0
Given Vector field is solenoidal
Given Vector field is irrotational
```

Output:

$f = i(4z + 2y + x) + k(2z - y + 4x) + j(-z - 3y + 2x)$
 $div(f) = 0$
 $curl(f) = 0$
Given Vector field is solenoidal
Given Vector field is irrotational

Problem 6. Write a program to test whether $\vec{f} = (x^2 + xy^2)\hat{i} + (y^2 + x^2y)\hat{j}$ is solenoidal/irrotational or not

Program:

```
load ("vect")$
J:[i,j,k]$
f:[x^2+x*y^2,y^2+x^2*y, 0]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=",C.J)$
if D=0 then print("Given Vector field is solenoidal")
else print("Given Vector field is not solenoidal")$
if C.J=0 then print("Given Vector field is irrotational")
else print("Given Vector field is not irrotational")$
```

```
load ("vect")$
J:[i,j,k]$
f:[x^2+x*y^2,y^2+x^2*y, 0]$
D:ev(express(div(f)),diff)$
C:ev(express(curl(f)),diff)$
print("f=", f.J)$
print("div(f)=", D)$
print("curl(f)=",C.J)$
if D=0 then print("Given Vector field is solenoidal")
else print("Given Vector field is not solenoidal")$
if C.J=0 then print("Given Vector field is irrotational")
else print("Given Vector field is not irrotational")$
vect: warning: removing existing rule or rules for ".
f= i (x y^2+x^2)+j (y^2+x^2 y)
div(f)= y^2+2 y+x^2+2 x
curl(f)= 0
Given Vector field is not solenoidal
Given Vector field is irrotational
```

Output:

$$f = i(xy^2 + x^2) + j(y^2 + x^2y)$$

$$\text{div}(f) = y^2 + 2y + x^2 + 2x$$

$$\text{curl}(f) = 0$$

Given Vector field is not solenoidal

Given Vector field is irrotational

Exercise:

I. Write a program to find the directional derivative of Φ in the direction of $\vec{a}$ at the given point (x_0, y_0, z_0) .

1. $\Phi = yz + xz + xy, \vec{a} = 3i + 4j + 5k$ at $(1, 2, 3)$ (Ans: $\frac{23\sqrt{2}}{5}$)

2. $\Phi = x^3 + y^3 + z^3, \vec{a} = i + 2j + k$ at $(1, -1, 2)$ (Ans: $\frac{21}{\sqrt{6}}$)

II. Write a program to find the unit normal to the surface $\Phi = C$ at the given point (x_0, y_0, z_0) .

1. $x^3 + y^3 + 3xyz = 3$ at $(1, 2, -1)$ (Ans: $\frac{2k}{\sqrt{14}} + \frac{3j}{\sqrt{14}} - \frac{i}{\sqrt{14}}$)

2. $x^2y^2z^2 = 4$ at $(-1, -1, 2)$ (Ans: $\frac{k}{3} - \frac{2j}{3} - \frac{2i}{3}$)

III. Write a program to find the angle between surfaces $\Phi = C_1$ and $\Psi = C_2$ at the common point (x_0, y_0, z_0) .

1. $x^2 + y^2 + z^2 = 9$ and $x^2 + y^2 - z = 3$ at $(2, -1, 2)$ (Ans: $\cos^{-1}(\frac{8}{3\sqrt{21}})$)

2. $x \log(z) - y^2 = -1$ and $x^2y + z = 2$ at $(1, 1, 1)$ (Ans: $\pi - \cos^{-1}(\frac{1}{\sqrt{5}\sqrt{6}})$)

IV. Write a program to test whether given vector field $\vec{f}$ is solenoidal/irrotational or not.

1. $\vec{f} = (3x + 3y + 4z)\hat{i} + (x - 2y + 3z)\hat{j} + (3x + 2y - z)\hat{k}$

2. $\vec{f} = (x + 3y)\hat{i} + (y - 3z)\hat{j} + (x - 2z)\hat{k}$

3. $\vec{f} = (\sin(y) + z)\hat{i} + (x \cos(y) - z)\hat{j} + (x - y)\hat{k}$

4. $\vec{f} = x\hat{i} + y\hat{j} + z\hat{k}$

5. $\vec{f} = 3y^4z^2\hat{i} + 4x^3z^2\hat{j} - 3x^2y^2\hat{k}$

Experiment 6

Program 6: Program to evaluate vector line integral

Aim: To find $\int_c \vec{F} \cdot d\vec{r}$ which is work done in moving a particle in a force field where $\vec{F} = 2xy\vec{i} - 3x\vec{j} - 5z\vec{k}$, c is the curve $x = t, y = t^2 + 1, z = 2t^2$ from $t = 0$ to $t = 1$.

Software used: Maxima.

Definition:

Line integral: Let $\vec{F} = F_1\vec{i} + F_2\vec{j} + F_3\vec{k}$ be a vector point function in a region R and c is a regular curve from the point A to point B the line integral of $\vec{F}$ is defined as

$$\begin{aligned}\int_A^B \vec{F} \cdot d\vec{r} &= \int_A^B (F_1\vec{i} + F_2\vec{j} + F_3\vec{k}) \cdot (dx\vec{i} + dy\vec{j} + dz\vec{k}) \\ &= \int_A^B F_1 dx + F_2 dy + F_3 dz \quad \text{----- 1}\end{aligned}$$

1. The line integral (1) gives the total work done by force $\vec{F}$ along A to B.
2. If $x = x(t), y = y(t), z = z(t)$ are parametric equations of curve c , then

$$\int_A^B \vec{F} \cdot d\vec{r} = \int_{t=t_1}^{t_2} \left[F_1 \frac{dx}{dt} + F_2 \frac{dy}{dt} + F_3 \frac{dz}{dt} \right] dt$$

3. If c is a closed curve then $\oint \vec{F} \cdot d\vec{r}$ is called the circulation of $\vec{F}$ around

c . If $\oint \vec{F} \cdot d\vec{r} = 0$ is irrotational vector

Conclusion:

Work done in moving a particle in a force field is $\int_C \vec{F} \cdot d\vec{r} = \frac{-21}{2}$

Find the work done in moving a particle in a force field given by $\vec{F} = 2xy\vec{i} - 3x\vec{j} - 5z\vec{k}$ along the curve c given by $x = t, y = t^2 + 1, z = 2t^2$ from $t = 0$ to $t = 1$

PROGRAM:


```

/. Program 6: Program to evaluate vector line integral ./
kill(all)$
load("vect")$
f:[2·x·y,-3·x,-5·z]$
print("The given vector is:",f)$
f1:subst([x=t,y=t^2+1,z=2·t^2],f)$
print("f after substitution:",f1)$
x:t$
y:t^2+1$
z:2·t^2$
dx:diff(x,t)$
dy:diff(y,t)$
dz:diff(z,t)$
dr:[dx,dy,dz]$
fd:f1.dr$
print("Dot product is:",fd)$
li:integrate(fd,t,0,1)$
print("The vector line integral is",li)$

```

OUTPUT:

The given vector is: $[2xy, -3x, -5z]$
 f after substitution: $[2t(t^2+1), -3t, -10t^2]$
 Dot product is: $-40t^3 + 2t(t^2+1) - 6t^2$
 The vector line integral is $-\frac{21}{2}$

Experiment 7

Program 7: Program to evaluate the surface integral.

Aim: To evaluate $\iint_S \vec{F} \cdot \hat{n} \, ds$ where S is the surface $2x + 2y + z = 6$ and $\vec{F} = xy\mathbf{i} - x^2\mathbf{j} + (x + z)\mathbf{k}$.

Software used: maxima

Definition:

Surface integral:

An integral which is evaluated over the surface S is called the surface integral.

Let $\vec{F} = F_1\mathbf{i} + F_2\mathbf{j} + F_3\mathbf{k}$ be a vector point function and S is any surface in 3D-space then the surface integral is defined as $\iint_S \vec{F} \cdot \hat{n} \, ds$ where $\hat{n}$ is the unit

normal vector to surface S and $\hat{n} = \frac{\nabla\phi}{|\nabla\phi|}$

1. If R is the projection of surface S on xy -plane then

$$\iint_S \vec{F} \cdot \hat{n} \, ds = \iint_R \vec{F} \cdot \hat{n} \frac{dx dy}{|n \cdot k|} \text{ where } ds = dx dy$$

2. If R_1 is the projection of S on yz - plane then

$$\iint_S \vec{F} \cdot \hat{n} \, ds = \iint_{R_1} \vec{F} \cdot \hat{n} \frac{dy dz}{|n \cdot i|}$$

3. If R_2 is the projection of S on xz -plane then

$$\iint_S \vec{F} \cdot \hat{n} \, ds = \iint_{R_2} \vec{F} \cdot \hat{n} \frac{dx dz}{|n \cdot j|}$$

Conclusion:

$$\iint_S \vec{F} \cdot \hat{n} \, ds = \frac{27}{4}$$

Evaluate $\iint_S \vec{F} \cdot \hat{n} \, ds$ where S is the surface of the cylinder $2x + 2y + z = 6$ included in the first octant and $\vec{F} = xy\mathbf{i} - x^2\mathbf{j} + (x + z)\mathbf{k}$.

PROGRAM:

```

./ Program 7: Program to evaluate a surface integral ./
kill(all)$
load("vect")$
f:[x.y,-x.x,(x+z)]$
print("The given vector is",f)$
phi:2.x+2.y+z-6$
print("The given space is",phi)$
dphi:[diff(phi,x),diff(phi,y),diff(phi,z)]$
mag:sqrt(ratsimp(dphi[1]^2+dphi[2]^2)+dphi[3]^2)$
n:dphi/mag$
print("Unit vector n is",n)$
fn:f.n$
print("f.n is",fn)$
k:[0,0,1]$
nk:n.k$
print("n.k is",nk)$
z1:solve(phi,z)$
fint:subst(z1,fn)$
rat:fint/(n.k)$
s1:integrate(integrate(rat,y,0,3-x),x,0,3)$
print("The surface intergral is",s1)$

```

OUTPUT:

The given vector is $[xy, -x^2, z+x]$

The given space is $z+2y+2x-6$

Unit vector n is $[\frac{2}{3}, \frac{2}{3}, \frac{1}{3}]$

f.n is $\frac{z+x}{3} + \frac{2xy}{3} - \frac{2x^2}{3}$

n.k is $\frac{1}{3}$

The surface intergral is $\frac{27}{4}$

Experiment 8

Program 8: Program to evaluate the volume integral.

Aim: To evaluate $\iiint_V \nabla \cdot \vec{F} dV$ where $\vec{F} = (2x^2 - 3z)\mathbf{i} - 2xy\mathbf{j} - 4x\mathbf{k}$ where V is the closed region bounded by $x = 0$ $y = 0$ $z = 0$ and $2x + 2y + z = 4$

Software used: Maxima

Definition:

Volume integral:

Let $\vec{F} = F_1\mathbf{i} + F_2\mathbf{j} + F_3\mathbf{k}$ be a vector point and V be the volume. The volume integration of $\vec{F}$ over V is defined as

$$\iiint_V \vec{F} dV = \iiint_V (F_1\mathbf{i} + F_2\mathbf{j} + F_3\mathbf{k}) dx dy dz$$

The volume integral is also given by $\iiint_V \text{div } \vec{F} dV$

Conclusion: $\iiint_V \nabla \cdot \vec{F} dV = \frac{8}{3}$

If $\vec{F} = (2x^2 - 3z)\mathbf{i} - 2xy\mathbf{j} - 4x\mathbf{k}$ evaluate $\iiint_V \nabla \cdot \vec{F} dV$ where V is the closed region bounded by the planes $x = 0$ $y = 0$ $z = 0$ and $2x + 2y + z = 4$

PROGRAM:

If $\vec{F} = (2x^2 - 3z)\mathbf{i} - 2xy\mathbf{j} - 4x\mathbf{k}$ evaluate $\iiint_V \nabla \cdot \vec{F} dV$ where V is the closed region bounded by the planes $x = 0$ $y = 0$ $z = 0$ and $2x + 2y + z = 4$

```

/. Program 8: Program to evaluate a volume integral ./
kill(all)$
load("vect")$
f1:2·x^2-3·z$
f2:-2·x·y$
f3:-4·x$
f:[f1,f2,f3]$
print("The given vector is",f)$
df:diff(f1,x)+diff(f2,y)+diff(f3,z)$
print("del.f is",df)$
pl:2·x+2·y+z-4$
print("The given plane is",pl)$
z1:solve(pl,z);
y1:subst(z=0,pl)$
y2:solve(y1,y);
x1:subst(y=0,y1)$
x2:solve(x1,x);
vol_int:integrate(integrate(integrate(df,z,0,(-2·y-2·x+4)),y,0,2-x),x,0,2)$
print("Volume integral is",vol_int)$

```

OUTPUT:

The given vector is $[2x^2 - 3z, -2xy, -4x]$
 del.f is $2x$
 The given plane is $z + 2y + 2x - 4$
 $[z = -2y - 2x + 4]$
 $[y = 2 - x]$
 $[x = 2]$
 Volume integral is $\frac{8}{3}$

Experiment 10

Program 9: Program to verify Green's theorem .

Aim: To verify Greens theorem for $\oint (xy + y^2)dx + x^2dy$ where c is the closed curve bounded by $y = x$ and $y = x^2$.

Software used: Maxima

Green's theorem in the plane:

Statement: If $P(x, y)$ and $Q(x, y)$ be two continuous function having continuous partial derivatives in a region R of the xy -plane bounded by a simply closed curve c then

$$\oint_C Pdx + Qdy = \iint_R \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dxdy$$

1. Green's theorem gives a relation between the plane surface integral and the line integral.
2. The Green's theorem in vector form is given by

$$\oint_C \vec{F} \cdot d\vec{r} = \iint_R (\nabla \times \vec{F}) \cdot \vec{k} ds$$

3. Let $\vec{F} = Pi + Qj$ be irrotational in the region R of the xy -plane then

$$\text{Curl } F = \nabla \times \vec{F} = 0 \Rightarrow \oint_C \vec{F} \cdot d\vec{r} = 0$$

$\Rightarrow$ The work done by the force $\vec{F}$ in displaying a particle around the closed curve C is zero.

$\Rightarrow \vec{F}$ is conservative.

4. Suppose $\vec{F}$ is conservative force in the xy -plane then there exists a scalar point function ϕ such that $\vec{F} = \nabla \phi$

$$\Rightarrow \nabla \times \vec{F} = \nabla \times \nabla \phi = 0$$

$$\Rightarrow \oint_C \vec{F} \cdot d\vec{r} = 0$$

Conversely if $\oint_C \vec{F} \cdot d\vec{r} = 0$ then $\vec{F}$ is conservative .

Thus in the xy -plane the circulation of $\vec{F}$ around a closed path vanishes if and only if $\vec{F}$ is a conservative force.

Conclusion: Green's theorem is verified a given plane and a closed curve.

Verify Greens theorem for $\oint (xy + y^2)dx + x^2dy$ where c is the closed curve bounded by $y = x$ and $y = x^2$.


```

/* Program 9: Program to verify Green's Theorem */
kill(all)$
p:x*y+y^2$
q:x^2$
c1:y=x$
c2:y=x^2$
s1:solve(subst(y=x,c2));
y0:subst(x=s1[1],c1)$
y1:subst(x=s1[2],c1)$
print("The intersection of closed curve is (0,0) and (1,1)")$
print("Along c1:")$
eqp1:subst(y=x^2,p)$
eqq1:diff(x^2,x)$
int_c1:integrate(eqp1+q*eqq1,x,0,1)$
print("Integral value along c1:",int_c1)$
print("Along c2:")$
eqp2:subst(y=x,p)$
int_c2:integrate(eqp2+q,x,1,0)$
print("Integral value along c2:",int_c2)$
cl_cur:int_c1+int_c2$
print("Sum of integral values along curves c1 and c2 (LHS of Green's theorem):",cl_cur)$
dp:diff(p,y)$
dq:diff(q,x)$
cl_r:integrate(integrate(dq-dp,y,x^2,x),x,0,1)$
print("Integral value along the region R (RHS of Green's theorem):",cl_r)$
if cl_cur=cl_r then
    print("Green's theorem is verified")
else
    print("Green's theorem not verified")$

```

OUTPUT:

$[x=0, x=1]$

The intersection of closed curve is $(0,0)$ and $(1,1)$

Along c_1 :

Integral value along c_1 : $\frac{19}{20}$

Along c_2 :

Integral value along c_2 : -1

Sum of integral values along curves c_1 and c_2 (LHS of Green's theorem): $-\frac{1}{20}$

Integral value along the region R (RHS of Green's theorem): $-\frac{1}{20}$

Green's theorem is verified

Experiment 10

Program10: Program to find equations of sphere, cone and cylinder

Aim: Program to find equations of sphere, cone and cylinder.

Software used: Maxima

Sphere:

A sphere is the locus of a point which moves so that its distance from a fixed point always remains constant. The fixed point is called the center of the sphere and the constant distance is called radius of the sphere.

Equation of the sphere whose center is (a, b, c) and radius r is given by

$$(x - a)^2 + (y - b)^2 + (z - c)^2 = r^2$$

Equations of the sphere whose center is origin and radius r is given by

$$x^2 + y^2 + z^2 = r^2$$

The equation $x^2 + y^2 + z^2 + 2ux + 2vy + 2wz + d = 0$ represents a sphere whose center is $(-u, -v, -w)$ and radius $= \sqrt{u^2 + v^2 + w^2 - d}$

Condition for a sphere:

The given equation represents a sphere if

1. It is a second degree equation in x, y, z
2. Co-efficient of x^2 = co-efficient of y^2 = co-efficient of z^2
3. It does not contain the terms involving the products xy, yz and zx

Cone:

A cone is a solid whose surface is generated by a line passing through a fixed point and a fixed plane curve not containing the point, consisting of two equal sections joined at a point. The fixed point is called the vertex of the cone, the moving line in any position is called the generator and if the generator cuts a fixed curve then the curve is called the guiding curve.

The equation of the cone with vertex at origin is a homogeneous equation of second degree in x, y, z which is $ax^2 + by^2 + cz^2 + 2tyz + 2gzx + 2hxy = 0$.

A right circular cone is a surface generated by a straight line which passes through the fixed point. The fixed point is called the vertex the constant angle is called the semi-vertical angle and the fixed line through the fixed point is called axis of the cone.

The equation of right circular cone whose vertex as origin, axis- oz and semi-vertical angle α is given by $x^2 + y^2 = z^2 \tan^2 \alpha$

Cylinder:

- A. A cylinder is the surface generated by a straight line which is parallel to a fixed straight line and satisfies one more condition that it may intersect a fixed curve or touch a given surface.
- B. A cylinder whose equation is second degree in x, y, z is called a quadratic cylinder.
- C. A right circular cylinder is the surface generated by a straight line which is parallel to a fixed line is at a constant distance from it the fixed line is called the axis and the constant distance is called radius of the cylinder.

The equation of the right circular cylinder whose axis is z -axis and radius a is given by

$$(x - h)^2 + (y - k)^2 = a^2$$

Conclusion:

- 1. The equation of sphere with center at $(1, 2, -3)$ and radius $=4$ is given by

$$x^2 + y^2 + z^2 - 2x - 4y + 6z + 14 = 16$$

- 2. The equation of the cone with center at $(2, -3, 5)$ and semi-vertical angle

30° is given by
$$x^2 + y^2 - 4x + 6y + 13 = \frac{z^2 - 10z + 25}{3}$$

- 3. The equation of the cylinder with center at $(2, 3)$ and radius $=2$ is given by

$$x^2 + y^2 - 4x - 6y + 13 = 4$$

Find the equation of

- 1. Sphere with center at $(1, 2, -3)$ and radius $=4$
- 2. Cone with center at $(2, -3, 5)$ and semi-vertical angle 30°
- 3. Cylinder with center at $(2, 3)$ and radius $=2$

PROGRAM:

```

/. Program 10: Program to find the equation sphere, cone and cylinder ./
kill(all)$
load(ntrig)$
print("SPHERE EQUATION")$
a: 1$
b: 2$
c: -3$
r: 4$
sph_eqn: ratsimp((x - a)^2 + (y - b)^2 + (z - c)^2 = r^2)$
print("The sphere equation is",sph_eqn)$
print("CONE EQUATION")$
a:2$
b:-3$
c:5$
ang:30.*pi/180$
cone_eqn:ratsimp((x-a)^2+(y-b)^2=(z-c)^2*(tan(ang))^2)$
print("The cone equation is",cone_eqn)$
print("CYLINDER EQUATION")$
h:2$
k:3$
r:2$
cyl_eqn:ratsimp((x-h)^2+(y-k)^2=r^2)$
print("The cylinder equation is",cyl_eqn)$

```

OUTPUT:

SPHERE EQUATION

The sphere equation is $z^2 + 6z + y^2 - 4y + x^2 - 2x + 14 = 16$

CONE EQUATION

The cone equation is $y^2 + 6y + x^2 - 4x + 13 = \frac{z^2 - 10z + 25}{3}$

CYLINDER EQUATION

The cylinder equation is $y^2 - 6y + x^2 - 4x + 13 = 4$

Experiment 11

Program11: Program to find distance between a straight line and a plane.

Aim: To find the distance between the line $\frac{x-1}{2} = \frac{y-2}{4} = \frac{z+5}{2}$ and the plane

$$3x - 2y + z = 2$$

Software used: Maxima

Distance between straight line and a plane:

The distance between a straight line and plane can be found by finding distance between a point P on the straight line and the plane provided the line and plane are parallel. Consider a straight line

$$\frac{x-x_1}{l} = \frac{y-y_1}{m} = \frac{z-z_1}{n} \text{ -----(1)}$$

Where (x_1, y_1, z_1) are point on the line and l, m, n are direction vectors of the line and the plane $ax + by + cz + d = 0$ -----(2) where a, b, c are normal vector of the plane.

If $(li + mj + nk) \cdot (ai + bj + ck) = 0$ then the line and plane are parallel

The distance can be found by using $dist = \frac{ax_1 + by_1 + cz_1 + d}{\sqrt{a^2 + b^2 + c^2}}$

Conclusion: The line and plane are parallel and distance between line and plane is $\frac{4}{\sqrt{14}}$

Find the distance between the line $\frac{x-1}{2} = \frac{y-2}{4} = \frac{z+5}{2}$ and the plane $3x - 2y + z = 2$

PROGRAM:

```

/. Program 11: Program to find distance between a straight line and a plane ./
kill(all)$
load("vect")$
x1:1$
y1:2$
z1:-5$
print("Points on line",x1,y1,z1)$
u:2$
v:4$
w:2$
print("The directional vectors of line",u,v,w)$
a:3$
b:-2$
c:1$
d:2$
print("The normal vectors on plane",a,b,c)$
dv:[u,v,w]$
nv:[a,b,c]$
dot:dv.nv$
print("The dot product of directional vector and normal vector is",dot)$
if dot=0 then
(
    print("The line is parallel to the plane"),
    dist:abs(a·x1+b·y1+c·z1+d)/sqrt(a^2+b^2+c^2),
    print("The distance between line and plane is",dist)
)
else
    print("The line is not parallel to the plane and cannot find the the distance")$

```

OUTPUT:

Points on line 1 2 -5

The directional vectors of line 2 4 2

The normal vectors on plane 3 -2 1

The dot product of directional vector and normal vector is 0

The line is parallel to the plane

The distance between line and plane is $\frac{4}{\sqrt{14}}$

Experiment 12

Program 12: Program to construct and plot some standard surface.

Aim: To construct and plot ellipsoid, elliptic paraboloid and hyperbolic paraboloid.

Software used: Maxima

Standard surfaces:

1. Ellipsoid: Ellipsoid is a closed surface of which all plane cross sections are either ellipse or circle. An ellipsoid is symmetrical about three mutually perpendicular axes that intersect at the center.

If a, b, c are the principle semi-axes the general equation is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

If $a = b = c$ then the surface is a sphere and the intersection with any plane passing through it is a circle

- It is symmetrical about each of the co-ordinate planes as only even powers of x, y, z occur in its equation.
- It meets x -axis at $A(a, 0, 0), A'(-a, 0, 0)$;
 y -axis at $B(0, b, 0), B'(0, -b, 0)$;
 z -axis at $C(0, 0, c), C'(0, 0, -c)$;
- Its section by the co-ordinate planes are ellipses. *i. e.*, the section by yz

plane ($x = 0$) is the ellipse $\frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$

2. Elliptic paraboloid: Elliptic paraboloid is a quadric surface whose vertical cross sections are parabolas while horizontal cross sections are ellipses.

The general form of the equation is $\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{2z}{c}$

where a, b are constants that dictate the level of curvature in the xy and yz planes respectively. If c is positive then the shape is in positive direction and if it is negative then shape is on negative direction.

- It is symmetrical about yz plane and zx plane as only even powers of x and y occur in the equation.
- It meets the axis at the origin only and touches the xy plane.

- Its the section by yz plane ($x = 0$) is a parabola $y^2 = \frac{2b}{c} z$
Its the section by zx plane ($y = 0$) is a parabola $x^2 = \frac{2a}{c} z$

Its the section by xy plane ($z = 0$) is the point ellipse $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 0$

3. Hyperbolic paraboloid: A hyperbolic paraboloid is a doubly ruled surface shape the general form is $\frac{x^2}{a^2} - \frac{y^2}{b^2} = \frac{2z}{c}$. In this position the hyperbolic

paraboloid opens downwards along x -axis and upwards along the y -axis.

- It is symmetrical about yz plane and zx plane as only even powers of x and y occur in the equation.
 - It meets the axis at the origin only and touches the xy plane.
 - Its the section by yz plane ($x = 0$) is a parabola $y^2 = -\frac{2b^2}{c}z$ Its the section by zx plane ($y = 0$) is a parabola $x^2 = \frac{2a^2}{c}z$
- Its the section by xy plane ($z = 0$) is the part of lines $y = \pm \frac{b}{a}x$

Conclusion:

1. Equation of ellipsoid with $a = 2, b = 1, c = 3$ is

$$z = -\frac{3\sqrt{-4y^2-x^2+4}}{2}, \quad z = \frac{3\sqrt{-4y^2-x^2+4}}{2}$$

2. Equation of elliptic paraboloid with $a = 2, b = 3, c = 5$ is

$$\frac{4y^2+9x^2}{36} = \frac{2z}{5}$$

3. Equation of hyperbolic paraboloid with $a = 2, b = 3, c = 1$ is

$$\frac{x^2}{4} - \frac{y^2}{9} = 2z$$

Find the equation of

1. Ellipsoid with $a = 2, b = 1, c = 3$
2. Elliptic paraboloid with $a = 2, b = 3, c = 5$
3. Hyperbolic paraboloid with $a = 2, b = 3, c = 1$

PROGRAM:


```

/* Program 12: Program to construct and plot some standard surfaces */
kill(all)$
load(draw)$
print("ELLIPSOID")$
a: 2$
b: 1$
c: 3$
eq: (x^2/a^2) + (y^2/b^2) + (z^2/c^2) - 1$
zsol: solve(eq, z)$
print("The equation of ellipsoid is",zsol)$
draw3d
(
    explicit(zsol[1], x, -a, a, y, -b, b),[x, y, z],[x, -a, a],[y, -b, b],[z, -c, c],box = false,title = "Ellipsoid"
);
print("ELLIPTIC PARABOLOID")$
a1:2$
b1:3$
c1:5$
eq_ep:ratsimp((x^2/a1^2)+(y^2/b1^2)=(2*z)/c1)$
print("The equation of the elliptic paraboloid is",eq_ep)$
draw3d
(
    explicit(eq_ep, [x, y, z], domain = [[ - 3*a1, 3*a1], [- 3*b1, 3*b1], [ - 3*c1, 3*c1]]),[x, y, z],grid = 50,
    xlabel = "x", ylabel = "y",zlabel = "z",title = "Elliptic Paraboloid"
);
print("HYPERBOLIC PARABOLOID")$
a2: 2$
b2: 3$
c2: 1$
eq_hp: (x^2/a2^2)-(y^2/b2^2) = 2*z/c2$
print("The equation of Hyperbolic paraboloid is",eq_hp)$
draw
(
    implicit(eq_hp, [x, y, z], domain = [[ -3*a, 3*a], [ -3*b, 3*b], [- 3, 3]]),[x, y, z],grid = 50,
    xlabel = "x",ylabel = "y",zlabel = "z",title = "Hyperbolic Paraboloid"
);

```

OUTPUT:

ELLIPSOID

The equation of ellipsoid is $[z = -\frac{3\sqrt{-4y^2 - x^2 + 4}}{2}, z = \frac{3\sqrt{-4y^2 - x^2 + 4}}{2}]$

ELLIPTIC PARABOLOID

The equation of the elliptic paraboloid is $\frac{4y^2 + 9x^2}{36} = \frac{2z}{5}$

HYPERBOLIC PARABOLOID

The equation of Hyperbolic paraboloid is $\frac{x^2}{4} - \frac{y^2}{9} = 2z$



BLDEA'S S.B. ARTS AND K.C.P. SCIENCE

COLLEGE, VIJAYAPUR

(Affiliated to Rani Channamma University, Belagavi)

Practical's on DSC I
LINEAR ALGEBRA

(AS PER NEP-2020)

MANUAL

B. Sc. VI Semester

Name:.....

Class:.....

UUCMS No:.....

DEPARTMENT OF MATHEMATICS

CONTENTS

Sl. No.	Title	Page No
1	Contents	2
2	List of Programs	3
3	Program 1: Program on Linear Combination of vectors.	4
4	Program 2: Program to verify Linear Dependence and Independence.	12
3	Program 3: Program to find Basis and Dimension of the subspaces.	19
6	Program 4: Program to verify if a function is Linear Transformation or not.	27
7	Program 5: Program to find the Matrix of linear transformation.	33
8	Program 6: Program to find the Eigenvalues and Eigenvectors of a given linear transformation.	42
9	Program 7: Program on Rank – Nullity theorem.	50
10	Program 8: Program to verify if the given linear transformation is Singular / Non-singular.	59
11	Program 9: Program to find the Minimal Polynomial of given transformation.	65
12	Program 10: Program to find the Algebraic Multiplicity of the Eigenvalues of the given linear transformation.	72
13	Program 11: Program on Diagonalization.	81

List of Programs

For Sixth Semester DSC 1 Mathematics

(Practicals on Linear Algebra)

(4 Hours per Week and 56 hours per Semester)

1. Program on linear combination of vectors.
2. Program to verify linear dependence and independence.
3. Program to find basis and dimension of the subspaces.
4. Program to verify if a function is linear transformation or not.
5. Program to find the matrix of linear transformation.
6. Program to find the Eigenvalues and Eigenvectors of a given linear transformation.
7. Program on Rank – Nullity theorem.
8. Program to verify if the given linear transformation is singular/non-singular.
9. Program to find the minimal polynomial of given transformation.
10. Program to find the algebraic multiplicity of the Eigenvalues of the given linear transformation.
11. Program on diagonalization.

Program 1

Program on linear combination of vectors.

Aim: To check whether given vector is in the Linear Span of given subset of vectors of a vector space and expressing it as a linear combination of those vectors using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
. (dot)	The operator . represents noncommutative multiplication and scalar product. It is used for usual multiplication of matrices.
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
linsolve (A,B)	Solves the list of simultaneous linear equations for the list A of variables in list B. The expressions must each be polynomials in the variables and may be equations.
%r, %r1, %r2, etc	These are system generated symbols for arbitrary parameters for solution of under-determined system of equations
setify (A)	Constructs a set from the elements of the list A.
'	The single quote operator ' prevents evaluation.
load("ratpow")	loads the package ratpow which provides functions that return the coefficients of the numerator of a CRE polynomial in a given variable.
ratp_dense_coeffs (expr, x)	Returns the coefficients of powers of x from highest to lowest
matrix(R ₁ ,R ₂ ,...R _m)\$	Creates matrix whose rows are R ₁ ,R ₂ ,...R _m which are lists of equal length.
list_matrix_entries (M)	Returns a list containing the elements of the matrix M
radcan (expr)	Simplifies expr, which can contain logs, exponentials, and radicals.
and	The logical conjunction operator.
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print ("text", expr)\$	Displays text within inverted commas and evaluates and displays expr

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Linear Combination of vectors: Let V be a vector space over a field F . Let $v_1, v_2, \dots, v_n$ be any n vectors of V . A vector of the form

$$\sum_{i=1}^n \alpha_i v_i = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n$$

where $\alpha_1, \alpha_2, \dots, \alpha_n \in F$ are scalars, is called a linear combination of the vectors $v_1, v_2, \dots, v_n$. Any given vector v is a linear combination of vectors $v_1, v_2, \dots, v_n$ if $\exists$ scalars $\alpha_1, \alpha_2, \dots, \alpha_n \in F$ such that $v = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n$. Otherwise, v is not a linear combination of vectors $v_1, v_2, \dots, v_n$.

Linear Span of a subset: Let V be a vector space over a field F . Let $S = \{v_1, v_2, \dots, v_n\}$ be a non-empty subset of n vectors of V . The set of all linear combinations of elements of S is called its linear span and is denoted by $L(S)$.

$$L(S) = \left\{ \sum_{i=1}^n \alpha_i v_i = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n \mid v_i \in S, \alpha_i \in F, \forall i = 1, 2, \dots, n \right\}$$

Standard Vector spaces:

- i) $R^n = \{(x_1, x_2, \dots, x_n) \mid x_i \in R\}$ the set of all n -tuples of real numbers is a vector space over R
- ii) $P = \{\sum_{i=0}^n a_i x^i \mid n \text{ is a positive integer, } a_i \in R\}$ the set of all polynomials in x with real coefficients, at most of degree n is a vector space over R
- iii) $R^{m \times n} = \{[a_{ij}]_{m \times n} \mid a_{ij} \in R\}$ is the set of all real matrices of order $m \times n$ is a real vector space.

Program:

Program to check whether v is in $L(S)$. Expressing v as linear combination of vectors of S if it is in $L(S)$. where $S \subseteq R^n$

```
v: given vector$
v1: vector 1 of S$
v2: vector 2 of S$
v3: vector 3 of S$
S:[v1,v2,v3]$
a:[α,β,γ]$
M:linsolve(S.a-v,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)")
and print(" Required Linear combination is v=", M,['v1','v2','v3'])$
```

Program:

Program to check whether v is in $L(S)$. Expressing v as linear combination of vectors of S if it is in $L(S)$. where $S \subseteq P$

```
load("ratpow")$
v: given vector$
v1: vector 1 of S$
v2: vector 2 of S$
v3: vector 3 of S$
S:[v1,v2,v3]$
a:[ $\alpha,\beta,\gamma$ ]$
L:ratp_dense_coeffs(S.a-v,x)$
M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$
```

Program:

Program to check whether v is in $L(S)$. Expressing v as linear combination of vectors of S if it is in $L(S)$. where $S \subseteq R^{m \times n}$

```
v: given matrix$
v1: matrix 1 of S$
v2: matrix 2 of S$
v3: matrix 3 of S$
S:[v1,v2,v3]$
a:[ $\alpha,\beta,\gamma$ ]$
L:list_matrix_entries (S.a-v)$
M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$
```

Note: Number of members in list S and list a taken 3 each in above programs. Take actual number of members present in S of given problem.

Worked Examples:

Problem 1. Check whether $v = (1,2,3,5)$ is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ where $S = \{(2,3,4,5), (-1, -2, -3, -4), (6,7,8,9)\} \subseteq R^4$

Program:

```
v:[1,2,3,5]$
v1:[2,3,4,5]$
v2:[-1,-2,-3,-4]$
v3:[6,7,8,9]$
S:[v1,v2,v3]$
a:[α,β,γ]$
M:linsolve(S.a-v,a)$
```

```
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M.['v1','v2','v3'])$
```

```
) v:[1,2,3,5]$
v1:[2,3,4,5]$
v2:[-1,-2,-3,-4]$
v3:[6,7,8,9]$
S:[v1,v2,v3]$
a:[α,β,γ]$
M:linsolve(S.a-v,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M.['v1','v2','v3'])$

solve: dependent equations eliminated: (3)
Given set of vectors is S= {[-1, -2, -3, -4], [2, 3, 4, 5], [6, 7, 8, 9]}
Given vector is v= [1, 2, 3, 5]
v is not in L(S)
```

Output:

```
Given set of vectors is S = {[-1, -2, -3, -4], [2, 3, 4, 5], [6, 7, 8, 9]}
Given vector is v = [1, 2, 3, 5]
v is not in L(S)
```

Problem 2. Check whether $v = (2,4,3)$ is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ where $S = \{(1, -1, 0), (0, 2, 1)\} \subseteq R^3$

Program:

```
v:[2,4,3]$
v1:[1,-1,0]$
v2:[0,2,1]$
S:[v1,v2]$
a:[α,β]$
M:linsolve(S.a-v,a)$
```

```
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M.['v1','v2'])$
```

```
v:[2,4,3]$
v1:[1,-1,0]$
v2:[0,2,1]$
S:[v1,v2]$
a:[α,β]$
M:linsolve(S.a-v,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M.['v1','v2'])$

solve: dependent equations eliminated: (3)
Given set of vectors is S= {[0, 2, 1], [1, -1, 0]}
Given vector is v= [2, 4, 3]
v is in L(S)
Required Linear combination is v= v2 β + v1 α = 3 v2 + 2 v1
```

Output:

Given set of vectors is $S = \{[0,2,1], [1,-1,0]\}$

Given vector is $v = [2,4,3]$

v is in $L(S)$

Required Linear combination is $v = v_2\beta + v_1\alpha = 3v_2 + 2v_1$

Problem 3. Check whether $v = (1, -2, 5)$ is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ where $S = \{(1,1,1), (1,2,3), (2, -1,1)\} \subseteq R^3$

Program:

```
v:[1,-2,5]$
```

```
v1:[1,1,1]$
```

```
v2:[1,2,3]$
```

```
v3:[2,-1,1]$
```

```
S:[v1,v2,v3]$
```

```
a:[α,β,γ]$
```

```
M:linsolve(S.a-v,a)$
```

```
print("Given set of vectors is S=",setify(S))$
```

```
print("Given vector is v=",v)$
```

```
if M=[] then print("v is not in L(S)")
```

```
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$
```

```
v:[1,-2,5]$
v1:[1,1,1]$
v2:[1,2,3]$
v3:[2,-1,1]$
S:[v1,v2,v3]$
a:[α,β,γ]$
M:linsolve(S.a-v,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$

Given set of vectors is S= {[1,1,1],[1,2,3],[2,-1,1]}
Given vector is v= [1,-2,5]
v is in L(S)
Required Linear combination is v= v3 γ+v2 β+v1 α=2 v3+3 v2-6 v1
```

Output:

Given set of vectors is $S = \{[1,1,1], [1,2,3], [2, -1,1]\}$

Given vector is $v = [1, -2, 5]$

v is in $L(S)$

Required Linear combination is $v = v_3\gamma + v_2\beta + v_1\alpha = 2v_3 + 3v_2 - 6v_1$

Problem 4. Check whether $v = 2x^3 + x^2 + 3x - 1$ is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ where $S = \{x, x^2 + 1, x^3 - 1\} \subseteq P$

Program:

```
load("ratpow")$
```

```
v:2*x^3+x^2+3*x-1$
```

```
v1:x$
```

```
v2:x^2+1$
```

```
v3:x^3-1$
```

```
S:[v1,v2,v3]$
```

```
a:[α,β,γ]$
```

```
L:ratp_dense_coeffs(S.a-v,x)$
```

```
load("ratpow")$
v:2*x^3+x^2+3*x-1$
v1:x$
v2:x^2+1$
v3:x^3-1$
S:[v1,v2,v3]$
a:[α,β,γ]$
L:ratp_dense_coeffs(S.a-v,x)$
M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$

solve: dependent equations eliminated: (4)
Given set of vectors is S= {x, x^2+1, x^3-1}

Given vector is v= 2 x^3+x^2+3 x-1
v is in L(S)
Required Linear combination is v= v3 γ+v2 β+v1 α=2 v3+v2+3 v1
```

```

M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$

```

Output:

Given set of vectors is $S = \{x, x^2 + 1, x^3 - 1\}$
Given vector is $v = 2x^3 + x^2 + 3x - 1$
v is in L(S)
Required Linear combination is $v = v_3\gamma + v_2\beta + v_1\alpha = 2v_3 + v_2 + 3v_1$

Problem 5. Check whether $v = x^2 + 4x - 3$ is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ where $S = \{x^2 - 2x + 5, 2x^2 - 3x, x + 3\} \subseteq P$

Program:

```

load("ratpow")$
v:x^2+4*x-3$
v1:x^2-2*x+5$
v2:x^2-3*x$
v3:x+3$
S:[v1,v2,v3]$
a:[α,β,γ]$
L:ratp_dense_coeffs(S.a-v,x)$
M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$

Given set of vectors is S= {x+3, x^2-3x, x^2-2x+5}

Given vector is v= x^2+4x-3
v is in L(S)
Required Linear combination is v= v3γ+v2β+v1α=19v3+13v2-12v1

```

Output:

Given set of vectors is $S = \{x + 3, x^2 - 3x, x^2 - 2x + 5\}$
Given vector is $v = x^2 + 4x - 3$
v is in L(S)
Required Linear combination is $v = v_3\gamma + v_2\beta + v_1\alpha = 19v_3 + 13v_2 - 12v_1$

Problem 6. Check whether $v = \begin{bmatrix} 3 & -1 \\ 1 & -2 \end{bmatrix}$ is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ where $S = \left\{ \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \right\} \subseteq R^{2 \times 2}$

Program:

```
v:matrix([3,-1],[1,-2])$
```

```
v1:matrix([1,1],[0,-1])$
```

```
v2:matrix([1,1],[-1,0])$
```

```
v3:matrix([1,-1],[0,0])$
```

```
S:[v1,v2,v3]$
```

```
a:[α,β,γ]$
```

```
L:list_matrix_entries (S.a-v)$
```

```
M:linsolve(L,a)$
```

```
print("Given set of vectors is S=",setify(S))$
```

```
print("Given vector is v=",v)$
```

```
if M=[] then print("v is not in L(S)")
```

```
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$
```

```
v:matrix([3,-1],[1,-2])$
v1:matrix([1,1],[0,-1])$
v2:matrix([1,1],[-1,0])$
v3:matrix([1,-1],[0,0])$
S:[v1,v2,v3]$
a:[α,β,γ]$
L:list_matrix_entries (S.a-v)$
M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$
solve: dependent equations eliminated: (4)

Given set of vectors is S=  $\left\{ \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix} \right\}$ 

Given vector is v=  $\begin{pmatrix} 3 & -1 \\ 1 & -2 \end{pmatrix}$ 

v is in L(S)
Required Linear combination is v= v3 γ + v2 β + v1 α = 2 v3 - v2 + 2 v1
```

Output:

Given set of vectors is $S = \left\{ \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} \right\}$

Given vector is $v = \begin{bmatrix} 3 & -1 \\ 1 & -2 \end{bmatrix}$

v is in $L(S)$

Required Linear combination is $v = v_3 \gamma + v_2 \beta + v_1 \alpha = 2v_3 - v_2 + 2v_1$

Problem 7. Check whether $v = \begin{bmatrix} 6 & -1 \\ -8 & -8 \end{bmatrix}$ is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ where $S = \left\{ \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 2 & 4 \end{bmatrix}, \begin{bmatrix} 4 & -2 \\ 0 & -2 \end{bmatrix} \right\} \subseteq R^{2 \times 2}$

Program:

```
v:matrix([6,-1],[-8,-8])$
```

```
v1:matrix([1,2],[-1,3])$
```

```
v2:matrix([0,1],[2,4])$
```

```
v3:matrix([4,-2],[0,-2])$
```

```
S:[v1,v2,v3]$
```

```
a:[α,β,γ]$
```

```
L:list_matrix_entries (S.a-v)$
```

```

M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$

```

Output:

Given set of vectors is $S = \left\{ \begin{bmatrix} 0 & 1 \\ 2 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}, \begin{bmatrix} 4 & -2 \\ 0 & -2 \end{bmatrix} \right\}$

Given vector is $v = \begin{bmatrix} 6 & -1 \\ -8 & -8 \end{bmatrix}$

v is in $L(S)$

Required Linear combination is $v = v_3\gamma + v_2\beta + v_1\alpha = v_3 - 3v_2 + 2v_1$

```

v:matrix([6,-1],[-8,-8])$
v1:matrix([1,2],[-1,3])$
v2:matrix([0,1],[2,4])$
v3:matrix([4,-2],[0,-2])$
S:[v1,v2,v3]$
a:[α,β,γ]$
L:list_matrix_entries (S.a-v)$
M:linsolve(L,a)$
print("Given set of vectors is S=",setify(S))$
print("Given vector is v=",v)$
if M=[] then print("v is not in L(S)")
else print("v is in L(S)") and print(" Required Linear combination is v=", M,['v1','v2','v3'])$
solve: dependent equations eliminated: (4)

Given set of vectors is  $S = \left\{ \begin{pmatrix} 0 & 1 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ -1 & 3 \end{pmatrix}, \begin{pmatrix} 4 & -2 \\ 0 & -2 \end{pmatrix} \right\}$ 

Given vector is  $v = \begin{pmatrix} 6 & -1 \\ -8 & -8 \end{pmatrix}$ 

v is in L(S)
Required Linear combination is  $v = v_3\gamma + v_2\beta + v_1\alpha = v_3 - 3v_2 + 2v_1$ 

```

Exercise:

Write a program to check whether v is in $L(S)$ or not and express it as a linear combination of vectors of S if it is in $L(S)$ for the following:

1. $v = (3, -7, 6)$, $S = \{(1, -3, 2), (2, 4, 1), (1, 1, 1)\}$ (Answer: v is in $L(S)$)

2. $v = (3, 9, -4, 4)$, $S = \{(1, -2, 0, 3), (2, -1, 2, 1), (2, 3, 0, 1)\}$ (Answer: v is in $L(S)$)

3. $v = x^2 + x + 1$, $S = \{x, x^2 + 1, x^3 - 1\}$ (Answer: v is in $L(S)$)

4. $v = 2x^3 + x^2 - x - 5$, $S = \{1, x + 1, x^2 + x + 1, x^3 + x^2 + x + 1\}$ (Answer: v is in $L(S)$)

5. $v = \begin{bmatrix} -1 & 7 \\ 5 & 1 \end{bmatrix}$, $S = \left\{ \begin{bmatrix} 0 & 1 \\ 2 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}, \begin{bmatrix} 4 & -2 \\ 0 & -2 \end{bmatrix} \right\}$ (Answer: v is not in $L(S)$)

6. $v = \begin{bmatrix} 6 & 3 \\ 0 & 8 \end{bmatrix}$, $S = \left\{ \begin{bmatrix} 0 & 1 \\ 2 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}, \begin{bmatrix} 4 & -2 \\ 0 & -2 \end{bmatrix} \right\}$ (Answer: v is in $L(S)$)

Program 2

Program to verify linear dependence and independence

Aim: To check whether given set of vectors is linearly dependent or independent using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
. (dot)	The operator . represents noncommutative multiplication and scalar product. It is used for usual multiplication of matrices.
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
hipow (expr, x)	Returns the highest explicit exponent of x in expr
lopow (expr, x)	Returns the lowest exponent of x which explicitly appears in expr.
matrix(R ₁ , R ₂ , ..., R _m)\$	Creates matrix whose rows are R ₁ , R ₂ , ..., R _m which are lists of equal length.
list_matrix_entries (M)	Returns a list containing the elements of the matrix M
apply('matrix, L)	Converting nested lists L to matrix
rank (M)	Computes the rank of the matrix M.
setify (A)	Constructs a set from the elements of the list A.
'	The single quote operator ' prevents evaluation.
makelist (expr, i, i ₀ , i _{max})	Returns the list of elements obtained when ev (expr, i=j) is applied to the elements j of the sequence: i ₀ , i ₀ + 1, i ₀ + 2, ..., with j less than or equal to i _{max} .
length (expr)	Returns the number of parts in the external (displayed) form of expr
:=	The function definition operator
and	The logical conjunction operator.
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print ("text", expr)\$	Displays text within inverted commas and evaluates and displays expr

- Note:1. Press Shift+Enter for evaluation of commands and display of output.
2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
3. Replace dollar (\$) by semicolon (;) to see output of any input line.
4. Start each session with kill(all)\$ or quit())\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Linearly Dependent Set: Let V be a vector space over a field F . Let $S = \{v_1, v_2, \dots, v_n\}$ be a non-empty subset of n vectors of V . Then the set S is called linearly dependent if $\exists$ scalars $\alpha_1, \alpha_2, \dots, \alpha_n \in F$ such that

$$\sum_{i=1}^n \alpha_i v_i = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n = \mathbf{0} \text{ with } \alpha_i \neq \mathbf{0} \text{ for some } i$$

Linearly Independent Set: Let V be a vector space over a field F . Let $S = \{v_1, v_2, \dots, v_n\}$ be a non-empty subset of n vectors of V . Then the set S is called linearly independent iff

$$\sum_{i=1}^n \alpha_i v_i = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n = \mathbf{0} \Rightarrow \alpha_i = \mathbf{0} \forall i$$

Note that linear dependence and independence are mutually exclusive. Thus, a set is either linearly independent or dependent but not both. So, we can say that linearly dependent set is not linearly independent and vice-versa.

Standard Vector spaces:

- i) $R^n = \{(x_1, x_2, \dots, x_n) \mid x_i \in R\}$ the set of all n -tuples of real numbers is a vector space over R
- ii) $P = \{\sum_{i=0}^n a_i x^i \mid n \text{ is a positive integer, } a_i \in R\}$ the set of all polynomials in x with real coefficients, at most of degree n is a vector space over R
- iii) $R^{m \times n} = \{[a_{ij}]_{m \times n} \mid a_{ij} \in R\}$ is the set of all real matrices of order $m \times n$ is a real vector space.

Program:

Program to check whether $S = \{v_1, v_2, v_3\}$ is linearly dependent or independent where $S \subseteq R^n$

```
v1: vector 1 of S$
v2: vector 2 of S$
v3: vector 3 of S$
S:[v1,v2,v3]$
M:apply('matrix,S)$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

Note: In the above program, 3 vectors are taken in S for illustration. S can contain any number of vectors.

Program:

Program to check whether $S = \{v_1, v_2, v_3\}$ is linearly dependent or independent where $S \subseteq P$

```
v1: vector 1 of S$
v2: vector 2 of S$
v3: vector 3 of S$
S:[v1,v2,v3]$
L:makelist(makelist(coeff (S[j], x,i),i,lopow(S,x),hipow(S,x)),j,1,length(S))$
M:apply('matrix,L)$
print("Given set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

Note: In the above program, 3 vectors are taken in S for illustration. S can contain any number of vectors.

Program:

Program to check whether $S = \{v_1, v_2, v_3\}$ is linearly dependent or independent where $S \subseteq R^{m \times n}$

```
g(x):=list_matrix_entries(x)
v1: matrix 1 of S$
v2: matrix 2 of S$
v3: matrix 3 of S$
S:[v1,v2,v3]$
T:makelist(g(S[i]),i,length(S))$
M:apply('matrix,T)$
print("Given set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

Note: In the above program, 3 vectors are taken in S for illustration. S can contain any number of vectors.

Worked Examples:

Problem 1. Write a program to check whether $S = \{(1,0), (1,1)\}$ is linearly independent or not
where $S \subseteq R^2$

Program:

```
v1:[1,0]$
v2:[1,1]$
S:[v1,v2]$
M:apply('matrix,S)$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

```
v1:[1,0]$
v2:[1,1]$
S:[v1,v2]$
M:apply('matrix,S)$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$

Given Set of Vectors is S= {[1,0],[1,1]}
Given set of vectors is Linearly Independent
```

Output:

Given Set of Vectors is S = {[1,0], [1,1]}
Given set of vectors is Linearly Independent

Problem 2. Write a program to check whether $S = \{(1,1,2,4), (2, -1, -5,2), (1, -1, -4,0), (2,1,1,6)\}$ is linearly independent or not where $S \subseteq R^4$

Program:

```
v1:[1,1,2,4]$
v2:[2,-1,-5,2]$
v3:[1,-1,-4,0]$
v4:[2,1,1,6]$
S:[v1,v2,v3,v4]$
M:apply('matrix,S)$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

```
v1:[1,1,2,4]$
v2:[2,-1,-5,2]$
v3:[1,-1,-4,0]$
v4:[2,1,1,6]$
S:[v1,v2,v3,v4]$
M:apply('matrix,S)$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$

Given Set of Vectors is S= {[1,-1,-4,0],[1,1,2,4],[2,-1,-5,2],[2,1,1,6]}
Given set of vectors is Linearly Dependent
```

Output:

Given Set of Vectors is S = {[1, -1, -4,0], [1,1,2,4], [2, -1, -5,2], [2,1,1,6]}
Given set of vectors is Linearly Dependent

Problem 3. Write a program to check whether $S = \{(1,0,-1), (3,1,1), (0,1,-1)\}$ is linearly independent or not where $S \subseteq R^3$

Program:

```
v1:[1,0,-1]$
v2:[3,1,1]$
v3:[0,1,-1]$
S:[v1,v2,v3]$
M:apply('matrix,S)$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$

Given Set of Vectors is S= {[0, 1, -1], [1, 0, -1], [3, 1, 1]}
Given set of vectors is Linearly Independent
```

Output:

Given Set of Vectors is S = {[0,1, -1], [1,0, -1], [3,1,1]}
Given set of vectors is Linearly Independent

Problem 4. Write a program to check whether $S = \{x^2 + 1, x - 1, x + 1\}$ is linearly independent or not where $S \subseteq P$

Program:

```
v1:x^2+1$
v2:x-1$
v3:x+1$
S:[v1,v2,v3]$
L:makelist(makelist(coeff (S[j], x,i),i,lopow(S,x),hipow(S,x)),j,1,length(S))$
M:apply('matrix,L)$
print("Given set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$

Given set of Vectors is S= {x-1,x+1,x^2+1}
Given set of vectors is Linearly Independent
```

Output:

Given Set of Vectors is S = {x - 1, x + 1, x² + 1}
Given set of vectors is Linearly Independent

Problem 5. Write a program to check whether $S = \{1, 1+x, 1+x+x^2, 1+x+x^2+x^3\}$ is linearly independent or not where $S \subseteq P$

Program:

```
v1:1$
v2:1+x$
v3:1+x+x^2$
v4:1+x+x^2+x^3$
S:[v1,v2,v3,v4]$
L:makelist(makelist(coeff(S[j], x,i),i,lopow(S,x),hipow(S,x)),j,1,length(S))$
M:apply('matrix,L)$
print("Given set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

Given set of Vectors is $S = \{1, x+1, x^2+x+1, x^3+x^2+x+1\}$
 Given set of vectors is Linearly Independent

Output:

Given Set of Vectors is $S = \{1, x+1, x^2+x+1, x^3+x^2+x+1\}$
 Given set of vectors is Linearly Independent

Problem 6. Write a program to check whether $S = \left\{ \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 2 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 8 \\ -4 & 12 \end{bmatrix} \right\}$ is linearly independent or not where $S \subseteq R^{2 \times 2}$

Program:

```
g(x):=list_matrix_entries(x)$
v1:matrix([1,2],[-1,3])$
v2:matrix([0,1],[2,4])$
v3:matrix([4,8],[-4,12])$
S:[v1,v2,v3]$
T:makelist(g(S[i]),i,length(S))$
M:apply('matrix,T)$
print("Given set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

Given set of Vectors is $S = \left\{ \begin{pmatrix} 0 & 1 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ -1 & 3 \end{pmatrix}, \begin{pmatrix} 4 & 8 \\ -4 & 12 \end{pmatrix} \right\}$
 Given set of vectors is Linearly Dependent

Output:

Given Set of Vectors is $S = \left\{ \begin{pmatrix} 0 & 1 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ -1 & 3 \end{pmatrix}, \begin{pmatrix} 4 & 8 \\ -4 & 12 \end{pmatrix} \right\}$
 Given set of vectors is Linearly Dependent

Problem 7. Write a program to check whether $S = \left\{ \begin{bmatrix} 0 & 1 & -1 \\ 1 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ 1 & 2 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 1 \\ 0 & 3 & -1 \end{bmatrix} \right\}$ is linearly independent or not where $S \subseteq R^{2 \times 3}$

Program:

```
g(x):=list_matrix_entries(x)$
v1:matrix([1,2,1],[0,3,-1])$
v2:matrix([1,-1,1],[1,2,0])$
v3:matrix([0,1,-1],[1,2,1])$
S:[v1,v2,v3]$
T:makelist(g(S[i]),i,length(S))$
M:apply('matrix,T)$
print("Given set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$
```

```
g(x):=list_matrix_entries(x)$
v1:matrix([1,2,1],[0,3,-1])$
v2:matrix([1,-1,1],[1,2,0])$
v3:matrix([0,1,-1],[1,2,1])$
S:[v1,v2,v3]$
T:makelist(g(S[i]),i,length(S))$
M:apply('matrix,T)$
print("Given set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
else print("Given set of vectors is Linearly Dependent")$

Given set of Vectors is S=  $\left\{ \begin{pmatrix} 0 & 1 & -1 \\ 1 & 2 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 & 1 \\ 1 & 2 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 1 \\ 0 & 3 & -1 \end{pmatrix} \right\}$ 
Given set of vectors is Linearly Independent
```

Output:

Given Set of Vectors is $S = \left\{ \begin{pmatrix} 0 & 1 & -1 \\ 1 & 2 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 & 1 \\ 1 & 2 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 1 \\ 0 & 3 & -1 \end{pmatrix} \right\}$
 Given set of vectors is Linearly Independent

Exercise:

Write a program to check linear dependence and independence of given set S

1. $S = \{(-1,0,1), (1,0,1), (1,1,0)\} \subseteq R^3$ (Answer: Linearly Independent)
2. $S = \{(-2,1,3), (1,2,3), (3,1,0)\} \subseteq R^3$ (Answer: Linearly Dependent)
3. $S = \{(1, -2, 5), (2, 3, 1)\} \subseteq R^3$ (Answer: Linearly Independent)
4. $S = \{2, x + 1, 3x - 1, x^2\} \subseteq P$ (Answer: Linearly Dependent)
5. $S = \{x^2 + 6x + 5, 2x^2 + 4x + 3, x^2 - 2x + 1\} \subseteq P$ (Answer: Linearly Independent)
6. $S = \left\{ \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 3 \\ 1 & 7 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 2 & 4 \end{bmatrix}, \begin{bmatrix} 4 & -2 \\ 0 & -2 \end{bmatrix} \right\} S \subseteq R^{2 \times 2}$ (Answer: Linearly Dependent)
7. $S = \left\{ \begin{bmatrix} 1 & -2 & 5 \\ -1 & 3 & 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & 0 \end{bmatrix}, \begin{bmatrix} 4 & 2 & 3 \\ 1 & -2 & 1 \end{bmatrix} \right\} S \subseteq R^{2 \times 3}$ (Answer: Linearly Independent)

Program 3

Program to find basis and dimension of the subspaces.

Aim: To find basis and dimension of the given subspace of a vector space using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
args(M)	Converts a matrix M to a nested list
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
triangularize (M)	Returns the upper triangular form of the matrix M, as produced by Gaussian elimination
firstn (expr, count)	Returns the first count arguments of expr
matrix(R ₁ , R ₂ , ..., R _m)\$	Creates matrix whose rows are R ₁ , R ₂ , ..., R _m which are lists of equal length.
list_matrix_entries (M)	Returns a list containing the elements of the matrix M
apply('matrix, L)	Converting nested lists L to matrix
rank (M)	Computes the rank of the matrix M.
setify (A)	Constructs a set from the elements of the list A.
'	The single quote operator ' prevents evaluation.
makelist (expr, i, i ₀ , i _{max})	Returns the list of elements obtained when ev (expr, i=j) is applied to the elements j of the sequence: i ₀ , i ₀ + 1, i ₀ + 2, ..., with j less than or equal to i _{max} .
length (expr)	Returns the number of parts in the external (displayed) form of expr
:=	The function definition operator
and	The logical conjunction operator.
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print ("text", expr)\$	Displays text within inverted commas and evaluates and displays expr

- Note: 1. Press Shift+Enter for evaluation of commands and display of output.
2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
3. Replace dollar (\$) by semicolon (;) to see output of any input line.
4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Basis and Dimension of a Vector space: Let V be a vector space over a field F . A subset $B = \{v_1, v_2, \dots, v_n\} \subseteq V$ is called a basis of V if

- i) B is linearly independent and
- ii) B spans V

Furthermore, the number of vectors in a basis of V is called the dimension of V

Example: 1) The dimension of R^n , the vector space of n -tuples of real numbers over R , is n

2) The dimension of $R^{m \times n}$, the vector space of all $m \times n$ real matrices over R , is mn

3) The dimension of P_n , the vector space of all polynomials in x of degree at most n with real coefficients over R , is $n + 1$

Basis and Dimension of a Subspace: Let $S = \{v_1, v_2, \dots, v_n\}$ be a subset of vector space V . A basis for the subspace spanned by S is a linearly independent subset $B \subseteq S$ which spans the subspace. If S itself is linearly independent, then S itself is a basis and its cardinality is the dimension of the subspace. If S is linearly dependent, then any maximal linearly independent subset of S will be a basis for the subspace and its cardinality will be the dimension.

Program:

Program to find a basis and the dimension of the subspace spanned by the set $S = \{v_1, v_2, v_3\}$ where $S \subseteq R^n$

```
v1: vector 1 of S$
v2: vector 2 of S$
v3: vector 3 of S$
S:[v1,v2,v3]$
M:triangularize(apply('matrix,S))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(firstn(args(M),rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$
```

Note: In the above program, 3 vectors are taken in S for illustration. S can contain any number of vectors.

Program:

Program to find a basis and the dimension of the subspace spanned by the set $S = \{v_1, v_2, v_3\}$
where $S \subseteq R^{m \times n}$

```
g(x):=list_matrix_entries(x)$  
f(x):=matrix([x[1],x[2]], [x[3],x[4]])$  
v1: vector 1 of S$  
v2: vector 2 of S$  
v3: vector 3 of S$  
S:[v1,v2,v3]$  
T:makelist(g(S[i]),i,length(S))$  
M:triangularize(apply('matrix,T))$  
print("Given Set of Vectors is S=", setify(S))$  
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")  
and print("A Basis for subspace L(S) is B=", setify(S))  
and print("Dimension of subspace L(S) =", rank(M))  
else print("Given set of vectors is Linearly Dependent")  
and print("A Basis for subspace L(S) is B=",  
setify(makelist(f(firstn(args(M),rank(M))[i]),i,1,rank(M))))  
and print("Dimension of subspace L(S) =", rank(M))$
```

Note: In the above program, 3 vectors are taken in S for illustration. S can contain any number of vectors.

Worked Examples:

Problem 1. Write a program to find a basis and the dimension of the subspace spanned by the set

$$S = \{(1,2,3), (3,1,0), (-2,1,3)\} \subseteq R^3$$

Program:

```
v1:[1,2,3]$
v2:[3,1,0]$
v3:[-2,1,3]$
S:[v1,v2,v3]$
M:triangularize(apply('matrix,S))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(firstn(args(M),rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$
```

```
v1:[1,2,3]$
v2:[3,1,0]$
v3:[-2,1,3]$
S:[v1,v2,v3]$
M:triangularize(apply('matrix,S))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(firstn(args(M),rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$

Given Set of Vectors is S= {[ -2, 1, 3], [1, 2, 3], [3, 1, 0]}
Given set of vectors is Linearly Dependent
A Basis for subspace L(S) is B= {[0, 5, 9], [3, 1, 0]}
Dimension of subspace L(S) = 2
```

Output:

Given Set of Vectors is S = {[-2,1,3], [1,2,3], [3,1,0]}
Given set of vectors is Linearly Dependent
A Basis for subspace L(S) is B = {[0,5,9], [3,1,0]}
Dimension of subspace L(S) = 2

Problem 2. Write a program to find a basis and the dimension of the subspace spanned by the set

$$S = \{(1,1,2,4), (2, -1, -5,2), (1, -1, -4, 0), (2,1,1,6)\} \subseteq R^4$$

Program:

```
v1:[1,1,2,4]$
v2:[2,-1,-5,2]$
v3:[1,-1,-4,0]$
v4:[2,1,1,6]$
S:[v1,v2,v3,v4]$
M:triangularize(apply('matrix,S))$
print("Given Set of Vectors is S=", setify(S))$
```

```
v1:[1,1,2,4]$
v2:[2,-1,-5,2]$
v3:[1,-1,-4,0]$
v4:[2,1,1,6]$
S:[v1,v2,v3,v4]$
M:triangularize(apply('matrix,S))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(firstn(args(M),rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$

Given Set of Vectors is S= {[1, -1, -4, 0], [1, 1, 2, 4], [2, -1, -5, 2], [2, 1, 1, 6]}
Given set of vectors is Linearly Dependent
A Basis for subspace L(S) is B= {[0, 1, 3, 2], [1, -1, -4, 0]}
Dimension of subspace L(S) = 2
```



```

if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(firstn(args(M),rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$

```

Output:

Given Set of Vectors is $S = \{[1, -1, -4, 0], [1, 1, 2, 4], [2, -1, -5, 2], [2, 1, 1, 6]\}$
Given set of vectors is Linearly Dependent
A Basis for subspace $L(S)$ is $B = \{[0, 1, 3, 2], [1, -1, -4, 0]\}$
Dimension of subspace $L(S) = 2$

Problem 3. Write a program to find a basis and the dimension of the subspace spanned by the set

$$S = \{(1, 2, 0), (1, 1, 1), (2, 0, 1)\} \subseteq R^3$$

Program:

```

v1:[1,2,0]$
v2:[1,1,1]$
v3:[2,0,1]$
S:[v1,v2,v3]$
M:triangularize(apply('matrix,S))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(firstn(args(M),rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$

```

```

v1:[1,2,0]$
v2:[1,1,1]$
v3:[2,0,1]$
S:[v1,v2,v3]$
M:triangularize(apply('matrix,S))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(firstn(args(M),rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$

Given Set of Vectors is S= {[1, 1, 1],[1, 2, 0],[2, 0, 1]}
Given set of vectors is Linearly Independent
A Basis for subspace L(S) is B= {[1, 1, 1],[1, 2, 0],[2, 0, 1]}
Dimension of subspace L(S) = 3

```

Output:

Given Set of Vectors is $S = \{[1, 1, 1], [1, 2, 0], [2, 0, 1]\}$
Given set of vectors is Linearly Independent
A Basis for subspace $L(S)$ is $B = \{[1, 1, 1], [1, 2, 0], [2, 0, 1]\}$
Dimension of subspace $L(S) = 3$

Problem 4. Write a program to find a basis and the dimension of the subspace spanned by the set

$$S = \left\{ \begin{bmatrix} 1 & -5 \\ -4 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & 5 \end{bmatrix}, \begin{bmatrix} 2 & -4 \\ -5 & 7 \end{bmatrix}, \begin{bmatrix} 1 & -7 \\ -5 & 1 \end{bmatrix} \right\} \subseteq R^{2 \times 2}$$

Program:

```
g(x):=list_matrix_entries(x)$
```

```
f(x):=matrix([x[1],x[2]], [x[3],x[4]])$
```

```
v1:matrix([1,-5],[-4,2])$
```

```
v2:matrix([1,1],[-1,5])$
```

```
v3:matrix([2,-4],[-5,7])$
```

```
v4:matrix([1,-7],[-5,1])$
```

```
S:[v1,v2,v3,v4]$
```

```
T:makelist(g(S[i]),i,length(S))$
```

```
M:triangularize(apply('matrix,T))$
```

```
print("Given Set of Vectors is S=", setify(S))$
```

```
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
```

```
and print("A Basis for subspace L(S) is B=", setify(S))
```

```
and print("Dimension of subspace L(S) =", rank(M))
```

```
else print("Given set of vectors is Linearly Dependent")
```

```
and print("A Basis for subspace L(S) is B=", setify(makelist(f(firstn(args(M),rank(M)))[i]),i,1,rank(M))))
```

```
and print("Dimension of subspace L(S) =", rank(M))$
```

```
g(x):=list_matrix_entries(x)$
f(x):=matrix([x[1],x[2]], [x[3],x[4]])$
v1:matrix([1,-5],[-4,2])$
v2:matrix([1,1],[-1,5])$
v3:matrix([2,-4],[-5,7])$
v4:matrix([1,-7],[-5,1])$
S:[v1,v2,v3,v4]$
T:makelist(g(S[i]),i,length(S))$
M:triangularize(apply('matrix,T))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(makelist(f(firstn(args(M),rank(M)))[i]),i,1,rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$

Given Set of Vectors is S=  $\left\{ \begin{bmatrix} 1 & -7 \\ -5 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -5 \\ -4 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & 5 \end{bmatrix}, \begin{bmatrix} 2 & -4 \\ -5 & 7 \end{bmatrix} \right\}$ 

Given set of vectors is Linearly Dependent

A Basis for subspace L(S) is B=  $\left\{ \begin{bmatrix} 0 & 6 \\ 3 & 3 \end{bmatrix}, \begin{bmatrix} 1 & -5 \\ -4 & 2 \end{bmatrix} \right\}$ 

Dimension of subspace L(S) = 2
```

Output:

Given Set of Vectors is S = $\left\{ \begin{bmatrix} 1 & -7 \\ -5 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -5 \\ -4 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & 5 \end{bmatrix}, \begin{bmatrix} 2 & -4 \\ -5 & 7 \end{bmatrix} \right\}$

Given set of vectors is Linearly Dependent

A Basis for subspace L(S) is B = $\left\{ \begin{bmatrix} 0 & 6 \\ 3 & 3 \end{bmatrix}, \begin{bmatrix} 1 & -5 \\ -4 & 2 \end{bmatrix} \right\}$

Dimension of subspace L(S) = 2

Problem 5. Write a program to find a basis and the dimension of the subspace spanned by the set

$$S = \left\{ \begin{bmatrix} 1 & -2 & 5 \\ -1 & 3 & 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & 0 \end{bmatrix}, \begin{bmatrix} 2 & -3 & 6 \\ 1 & 6 & 2 \end{bmatrix} \right\} \subseteq R^{2 \times 3}$$

Program:

```
g(x):=list_matrix_entries(x)$
```

```
f(x):=matrix([x[1],x[2],x[3]], [x[4],x[5],x[6]])$
```

```
v1:matrix([1,-2,5],[-1,3,2])$
```

```
v2:matrix([1,-1,1],[2,3,0])$
```

```
v3:matrix([2,-3,6],[1,6,2])$
```

```
S:[v1,v2,v3]$
```

```
T:makelist(g(S[i]),i,length(S))$
```

```
M:triangularize(apply('matrix,T))$
```

```
print("Given Set of Vectors is S=", setify(S))$
```

```
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
```

```
and print("A Basis for subspace L(S) is B=", setify(S))
```

```
and print("Dimension of subspace L(S) =", rank(M))
```

```
else print("Given set of vectors is Linearly Dependent")
```

```
and print("A Basis for subspace L(S) is B=", setify(makelist(f(firstn(args(M),rank(M)))[i]),i,1,rank(M))))
```

```
and print("Dimension of subspace L(S) =", rank(M))$
```

```
g(x):=list_matrix_entries(x)$
f(x):=matrix([x[1],x[2],x[3]], [x[4],x[5],x[6]])$
v1:matrix([1,-2,5],[-1,3,2])$
v2:matrix([1,-1,1],[2,3,0])$
v3:matrix([2,-3,6],[1,6,2])$
S:[v1,v2,v3]$
T:makelist(g(S[i]),i,length(S))$
M:triangularize(apply('matrix,T))$
print("Given Set of Vectors is S=", setify(S))$
if rank(M)=length(S) then print("Given set of vectors is Linearly Independent")
and print("A Basis for subspace L(S) is B=", setify(S))
and print("Dimension of subspace L(S) =", rank(M))
else print("Given set of vectors is Linearly Dependent")
and print("A Basis for subspace L(S) is B=", setify(makelist(f(firstn(args(M),rank(M)))[i]),i,1,rank(M))))
and print("Dimension of subspace L(S) =", rank(M))$
```

Given Set of Vectors is $S = \left\{ \begin{bmatrix} 1 & -2 & 5 \\ -1 & 3 & 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & 0 \end{bmatrix}, \begin{bmatrix} 2 & -3 & 6 \\ 1 & 6 & 2 \end{bmatrix} \right\}$

Given set of vectors is Linearly Dependent

A Basis for subspace $L(S)$ is $B = \left\{ \begin{bmatrix} 0 & -1 & 4 \\ -3 & 0 & 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & 0 \end{bmatrix} \right\}$

Dimension of subspace $L(S) = 2$

Output:

Given Set of Vectors is $S = \left\{ \begin{bmatrix} 1 & -2 & 5 \\ -1 & 3 & 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & 0 \end{bmatrix}, \begin{bmatrix} 2 & -3 & 6 \\ 1 & 6 & 2 \end{bmatrix} \right\}$

Given set of vectors is Linearly Dependent

A Basis for subspace $L(S)$ is $B = \left\{ \begin{bmatrix} 0 & -1 & 4 \\ -3 & 0 & 2 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & 0 \end{bmatrix} \right\}$

Dimension of subspace $L(S) = 2$

Exercise:

Write a program to find a basis and the dimension of the subspace spanned by the set S

$$1. S = \{(2,4,2), (1, -1,0), (1,2,1), (0,3,1)\} \subseteq R^3$$

(Answer: $B=\{(0,6,2), (1, -1,0)\}$, $\dim L(S)=2$)

$$2. S = \{(1,1,1), (1,0,1), (1,0,0), (0,0,1)\} \subseteq R^3$$

(Answer: $B=\{(0,0,1), (0,1,1), (0,0,1)\}$, $\dim L(S)=3$)

$$3. S = \{(1,1), (1, -1)\} \subseteq R^2$$

(Answer: $B=\{(1,1), (1, -1)\}$, $\dim L(S)=2$)

$$4. S = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right\} \subseteq R^{2 \times 2}$$

(Answer: $B=\left\{ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right\}$, $\dim L(S)=3$)

Program 4

Program to verify if a function is a linear transformation or not.

Aim: To verify if the given function is a linear transformation or not from a vector space to a vector space, using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
. (dot)	The operator . represents noncommutative multiplication and scalar product. It is used for usual multiplication of matrices.
* (asterisk)	The operator * represents commutative multiplication
^	Exponentiation operator or power or index
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
L[i]	Returns i-th element of the list L
'	The single quote operator ' prevents evaluation.
makelist (expr, i, i ₀ , i _{max})	Returns the list of elements obtained when ev (expr, i=j) is applied to the elements j of the sequence: i ₀ , i ₀ + 1, i ₀ + 2, ..., with j less than or equal to i _{max} .
length (expr)	Returns the number of parts in the external (displayed) form of expr
:=	The function definition operator
and	The logical conjunction operator.
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print ("text", expr)\$	Displays text within inverted commas and evaluates and displays expr
radcan (expr)	Simplifies expr, which can contain logs, exponentials, and radicals.
assoc (key, e)	assoc searches for key as the first part of an argument of e and returns the second part of the first match, if any.

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Linear Transformation from a vector space to a vector space: Let U and V be two vector spaces over the same field F . A mapping or function $T: U \rightarrow V$ is called a linear transformation if

$$T(\alpha u_1 + \beta u_2) = \alpha T(u_1) + \beta T(u_2) \quad \forall u_1, u_2 \in U \text{ and } \alpha, \beta \in F$$

A linear transformation is also called a linear mapping. It is a mapping between two vector spaces that preserves the operations of vector addition and scalar multiplication.

To verify whether given function T is a linear transformation or not, just verify that

$T(\alpha u_1 + \beta u_2)$ and $\alpha T(u_1) + \beta T(u_2)$ are equal. If these are not equal, then T is not a linear transformation.

Program:

Program to verify whether given function $T: R^m \rightarrow R^n$ is a linear transformation or not.

```
T(x):=define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
Y:[p,q,r]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function isT(x,y,z)=", T(X))$
print("T(aX+bY)=",LHS)$
print("aT(X)+bT(Y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$
```

Note: In the above program $X:[x,y,z]$, $Y:[p,q,r]$ is taken for illustration. Take X , Y according to given domain of T .

1. For $R^m = R^2$, take $X:[x,y]$, $Y:[p,q]$
2. For $R^m = R^3$, take $X:[x,y,z]$, $Y:[p,q,r]$
3. For $R^m = R^4$, take $X:[w,x,y,z]$, $Y:[p,q,r,s]$ etc.

Examples for defining function:

1. If $T(x, y, z) = (2x + 3y - z, 5z - y)$, then define

$$T(x):= [2 * x[1] + 3 * x[2] - x[3], 5 * x[3] - x[2]]$$

2. If $T(x, y) = (y - x, y, x + y)$, then define

$$T(x):= [x[2] - x[1], x[2], x[1] + x[2]] \text{ etc.}$$

Worked Examples:

Problem 1. Write a program to verify whether $T: \mathbf{R}^3 \rightarrow \mathbf{R}^2$ defined by $T(x, y, z) = (x + y, y + z)$ is a linear transformation or not.

Program:

```
T(x):=[x[1]+x[2],x[2]+x[3]]$
X:[x,y,z]$
Y:[p,q,r]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function isT(x,y,z)=", T(X))$
print("T(aX+bY)=",LHS)$
print("aT(X)+bT(Y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$
```

```
T(x):=[x[1]+x[2],x[2]+x[3]]$
X:[x,y,z]$
Y:[p,q,r]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function isT(x,y,z)=", T(X))$
print("T(aX+bY)=",LHS)$
print("aT(X)+bT(Y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$

Given function isT(x,y,z)= [y+x,z+y]
T(aX+bY)= [ay+ax+bq+bp,az+ay+br+bq]
aT(X)+bT(Y)= [ay+ax+bq+bp,az+ay+br+bq]
Given function is a Linear transformation
```

Output:

Given function is $T(x, y, z) = [y + x, z + y]$
 $T(aX + bY) = [ay + ax + bq + bp, az + ay + br + bq]$
 $aT(X) + bT(Y) = [ay + ax + bq + bp, az + ay + br + bq]$
Given function is a Linear transformation

Problem 2. Write a program to verify whether $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x + y - z, 2x - y, y + 2z)$ is a linear transformation or not.

Program:

```
T(x):=[x[1]+x[2]-x[3],2*x[1]-x[2],x[2]+2*x[3]]$
X:[x,y,z]$
Y:[p,q,r]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function isT(x,y,z)=", T(X))$
print("T(aX+bY)=",LHS)$
print("aT(X)+bT(Y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$
```

```
T(x):=[x[1]+x[2]-x[3],2*x[1]-x[2],x[2]+2*x[3]]$
X:[x,y,z]$
Y:[p,q,r]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function isT(x,y,z)=", T(X))$
print("T(aX+bY)=",LHS)$
print("aT(X)+bT(Y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$

Given function isT(x,y,z)= [-z+y+x,2x-y,2z+y]
T(aX+bY)= [-az+ay+ax-br+bq+bp,-ay+2ax-bq+2bp,2az+ay+2br+bq]
aT(X)+bT(Y)= [-az+ay+ax-br+bq+bp,-ay+2ax-bq+2bp,2az+ay+2br+bq]
Given function is a Linear transformation
```

Output:

Given function is $T(x, y, z) = [-z + y + x, 2x - y, 2z + y]$

$T(aX + bY) = [-az + ay + ax - br + bq + bp, -ay + 2ax - bq + 2bp, 2az + ay + 2br + bq]$

$aT(X) + bT(Y) = [-az + ay + ax - br + bq + bp, -ay + 2ax - bq + 2bp, 2az + ay + 2br + bq]$

Given function is a Linear transformation

Problem 3. Write a program to verify whether $T: \mathbf{R}^3 \rightarrow \mathbf{R}$ defined by $T(x, y, z) = \alpha x + \beta y + \gamma z$ for a fixed (α, β, γ) is a linear transformation or not.

Program:

```
T(x):=α*x[1]+β*x[2]+γ*x[3]$
```

```
X:[x,y,z]$
```

```
Y:[p,q,r]$
```

```
LHS:radcan(T(a*X+b*Y))$
```

```
RHS:radcan(a*T(X)+b*T(Y))$
```

```
print("Given function isT(x,y,z)=", T(X))$
```

```
print("T(aX+bY)=",LHS)$
```

```
print("aT(X)+bT(Y)=",RHS)$
```

```
if LHS=RHS then print("Given function is a Linear transformation")
```

```
else print("Given function is not a linear transformation")$
```

```
T(x):=α*x[1]+β*x[2]+γ*x[3]$
X:[x,y,z]$
Y:[p,q,r]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function isT(x,y,z)=", T(X))$
print("T(aX+bY)=",LHS)$
print("aT(X)+bT(Y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$
Given function isT(x,y,z)= z γ+y β+x α
T(aX+bY)= (a z+b r) γ+(a y+b q) β+(a x+b p) α
aT(X)+bT(Y)= (a z+b r) γ+(a y+b q) β+(a x+b p) α
Given function is a Linear transformation
```

Output:

Given function is $T(x, y, z) = z\gamma + y\beta + x\alpha$

$T(aX + bY) = (az + br)\gamma + (ay + bq)\beta + (ax + bp)\alpha$

$aT(X) + bT(Y) = (az + br)\gamma + (ay + bq)\beta + (ax + bp)\alpha$

Given function is a Linear transformation

Problem 4. Write a program to verify whether $T: \mathbf{R} \rightarrow \mathbf{R}^3$ defined by $T(x) = [x, x^2, x^3]$ is a linear transformation or not.

Program:

```
T(x):=[x,x^2,x^3]$
```

```
LHS:radcan(T(a*x+b*y))$
```

```
RHS:radcan(a*T(x)+b*T(y))$
```

```
print("Given function isT(x)=", T(x))$
```

```
print("T(ax+by)=",LHS)$
```

```
print("aT(x)+bT(y)=",RHS)$
```

```
if LHS=RHS then print("Given function is a Linear transformation")
```

```
else print("Given function is not a linear transformation")$
```

```
T(x):=[x,x^2,x^3]$
LHS:radcan(T(a*x+b*y))$
RHS:radcan(a*T(x)+b*T(y))$
print("Given function isT(x)=", T(x))$
print("T(ax+by)=",LHS)$
print("aT(x)+bT(y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$
Given function isT(x)= [ x , x^2 , x^3 ]
T(ax+by)= [ b y+a x , b^2 y^2+2 a b x y+a^2 x^2 , b^3 y^3+3 a b^2 x y^2+3 a^2 b x^2 y+a^3 x^3 ]
aT(x)+bT(y)= [ b y+a x , b y^2+a x^2 , b y^3+a x^3 ]
Given function is not a linear transformation
```


Output:

Given function is $T(x) = [x, x^2, x^3]$

$T(aX + bY) = [by + ax, b^2y^2 + 2abxy + a^2x^2, b^3y^3 + 3ab^2xy^2 + 3a^2bx^2y + a^3x^3]$

$aT(X) + bT(Y) = [by + ax, by^2 + ax^2, by^3 + ax^3]$

Given function is not a linear transformation

Problem 5. Write a program to verify whether $T: \mathbf{R}^4 \rightarrow \mathbf{R}^4$ defined by $T(w, x, y, z) = [0, x, y, z]$ is a linear transformation or not.

Program:

```
T(x):=[0,x[2],x[3],x[4]]$
```

```
X:[w,x,y,z]$
```

```
Y:[p,q,r,s]$
```

```
LHS:radcan(T(a*X+b*Y))$
```

```
RHS:radcan(a*T(X)+b*T(Y))$
```

```
print("Given function isT(w,x,y,z)=", T(X))$
```

```
print("T(aX+bY)=",LHS)$
```

```
print("aT(X)+bT(Y)=",RHS)$
```

```
if LHS=RHS then print("Given function is a Linear transformation")
```

```
else print("Given function is not a linear transformation")$
```

```
T(x):=[0,x[2],x[3],x[4]]$
X:[w,x,y,z]$
Y:[p,q,r,s]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function isT(w,x,y,z)=", T(X))$
print("T(aX+bY)=",LHS)$
print("aT(X)+bT(Y)=",RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$

Given function isT(w,x,y,z)= [0,x,y,z]
T(aX+bY)= [0,a x + b q, a y + b r, a z + b s]
aT(X)+bT(Y)= [0,a x + b q, a y + b r, a z + b s]
Given function is a Linear transformation
```

Output:

Given function is $T(w, x, y, z) = [0, x, y, z]$

$T(aX + bY) = [0, ax + bq, ay + br, az + bs]$

$aT(X) + bT(Y) = [0, ax + bq, ay + br, az + bs]$

Given function is a Linear transformation

Problem 6. Write a program to verify whether $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by

$T(x, y) = [x \cos(\theta) - y \sin(\theta), x \sin(\theta) + y \cos(\theta)]$ is a linear transformation or not.

Program:

```
T(x):=[x[1]*cos(theta)-x[2]*sin(theta),x[1]*sin(theta)+x[2]*cos(theta)]$
```

```
X:[x,y]$
```

```
Y:[p,q]$
```

```
LHS:radcan(T(a*X+b*Y))$
```

```
RHS:radcan(a*T(X)+b*T(Y))$
```

```
print("Given function isT(x,y)=", T(X))$
```

```
print("T(aX+bY)=",LHS)$
```

```
print("aT(X)+bT(Y)=",RHS)$
```

```

if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$

```

Output:

Given function is $T(x, y) = [x \cos(\theta) - y \sin(\theta), x \sin(\theta) + y \cos(\theta)]$

$T(aX + bY) = [(-ay - bq) \sin(\theta) + (ax + bp) \cos(\theta), (ax + bp) \sin(\theta) + (ay + bq) \cos(\theta)]$

$aT(X) + bT(Y) = [(-ay - bq) \sin(\theta) + (ax + bp) \cos(\theta), (ax + bp) \sin(\theta) + (ay + bq) \cos(\theta)]$

Given function is a Linear transformation

```

T(x):=[x[1]*cos(theta)-x[2]*sin(theta),x[1]*sin(theta)+x[2]*cos(theta)]$
X:[x,y]$
Y:[p,q]$
LHS:radcan(T(a*X+b*Y))$
RHS:radcan(a*T(X)+b*T(Y))$
print("Given function is T(x,y)=", T(X))$
print("T(aX+bY)=", LHS)$
print("aT(X)+bT(Y)=", RHS)$
if LHS=RHS then print("Given function is a Linear transformation")
else print("Given function is not a linear transformation")$

Given function is T(x,y)= [x cos(theta)-y sin(theta), x sin(theta)+y cos(theta)]

T(aX+bY)= [(-a y -b q) sin(theta)+(a x +b p) cos(theta), (a x +b p) sin(theta)+(a y +b q) cos(theta)]

aT(X)+bT(Y)= [(-a y -b q) sin(theta)+(a x +b p) cos(theta), (a x +b p) sin(theta)+(a y +b q) cos(theta)]

Given function is a Linear transformation

```

Exercise:

Write a program to verify whether given function is a linear transformation or not.

1. $T: \mathbb{R}^3 \rightarrow \mathbb{R}$ defined by $T(x, y, z) = y$ (Answer: linear transformation)
2. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (x, -y)$ (Answer: linear transformation)
3. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (x, y, z)$ (Answer: linear transformation)
4. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (x + y, y + z)$ (Answer: linear transformation)
5. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (x + y + z, x + y + z, x + y + z)$ (Answer: linear transformation)
6. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (0, 0)$ (Answer: linear transformation)
7. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (x + 1, y + 2)$ (Answer: not a linear transformation)
8. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (xy, yz)$ (Answer: not a linear transformation)
9. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined by $T(x, y) = (x, y, 5)$ (Answer: not a linear transformation)
10. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (x + y, 0)$ (Answer: linear transformation)

Program 5

Program to find the matrix of linear transformation.

Aim: To find the matrix of given linear transformation with respect to given bases using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
. (dot)	The operator . represents noncommutative multiplication and scalar product. It is used for usual multiplication of matrices.
* (asterisk)	The operator * represents commutative multiplication
$\wedge$	Exponentiation operator or power or index
$[a_1, a_2, \dots, a_m]$	List of numbers/objects $a_1, a_2, \dots, a_m$.
$L[i]$	Returns i-th element of the list L
'	The single quote operator ' prevents evaluation.
$\text{makelist}(expr, i, i_0, i_max)$	Returns the list of elements obtained when $ev(expr, i=j)$ is applied to the elements j of the sequence: $i_0, i_0 + 1, i_0 + 2, \dots$, with $ j $ less than or equal to $ i_max $.
$\text{length}(expr)$	Returns the number of parts in the external (displayed) form of $expr$
$:=$	The function definition operator
$\text{if cond_1 then expr_1 else expr_0}$	evaluates to $expr_1$ if cond_1 evaluates to true, otherwise the expression evaluates to $expr_0$.
$\text{print}("text", expr)\$$	Displays $text$ within inverted commas and evaluates and displays $expr$
$\text{radcan}(expr)$	Simplifies $expr$, which can contain logs, exponentials, and radicals.
$\text{linsolve}(A,B)$	Solves the list of simultaneous linear equations for the list A of variables in list B. The expressions must each be polynomials in the variables and may be equations.
$\text{transpose}(M)$	Returns the transpose of M .
$\text{assoc}(\text{key}, e)$	assoc searches for key as the first part of an argument of e and returns the second part of the first match, if any.
$\text{apply}(\text{matrix}, L)$	Converting nested lists L to matrix

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with $\text{kill(all)}\$$ or $\text{quit()}\$$ to remove previously assigned values of all symbols

Definitions and Formulae:

Matrix of a Linear Transformation: Let $T: U \rightarrow V$ be a linear transformation where U is a vector space of dimension n and V is a vector space of dimension m . Further, let $B_1 = \{e_1, e_2, \dots, e_n\}$ and $B_2 = \{f_1, f_2, \dots, f_m\}$ be ordered bases of U and V respectively. Then we have,

$$T(e_1) = a_{11}f_1 + a_{12}f_2 + a_{13}f_3 + \dots + a_{1m}f_m$$

$$T(e_2) = a_{21}f_1 + a_{22}f_2 + a_{23}f_3 + \dots + a_{2m}f_m$$

$$\vdots$$

$$T(e_n) = a_{n1}f_1 + a_{n2}f_2 + a_{n3}f_3 + \dots + a_{nm}f_m$$

These n equations form an $m \times n$ matrix which is transpose of coefficient matrix. In other words, an $m \times n$ matrix whose j -th column is the coefficients of j -th equation. This matrix is called the matrix of linear transformation T and is denoted by $[T]$. Thus,

$$[T] = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \dots & a_{nm} \end{bmatrix}$$

Program:

Program to find the matrix of linear transformation $T: R^n \rightarrow R^m$ with respect to standard bases $B_1 = \{e_1, e_2, \dots, e_n\}$ and $B_2 = \{f_1, f_2, \dots, f_m\}$

```
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
A: ordered standard basis of domain$
B: ordered standard basis of co-domain$
L: makelist(T(A[i]),i,1,length(A))$
M: transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$
```

Note: Take X according to domain and standard basis A of domain and B of codomain as:

1. $X:[x,y]$ if domain is R^2 and A or $B:[[1,0],[0,1]]$ for domain/codomain R^2
2. $X:[x,y,z]$ if domain is R^3 and A or $B:[[1,0,0],[0,1,0],[0,0,1]]$ for domain/codomain R^3
3. $X:[w,x,y,z]$ if domain is R^4 and A or $B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$ for domain/codomain R^4 so on.

Program to find the matrix of linear transformation $T: R^n \rightarrow R^m$ with respect to any bases

$B1 = \{a_1, a, \dots, a_n\}$ and $B2 = \{b_1, b_2, \dots, b_m\}$

```
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
A: ordered list of given basis of domain$
B: ordered list of given basis of co-domain$
a:[α,β]$
K: makelist(linsolve(B.a-T(A[i]),a),i,1,length(A))$
L: makelist(makelist(assoc(a[i],K[j]),i,1,length(a)),j,1,length(A))$
M: transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$
```

Note: 1. $X:[x,y,z]$ and $a:[\alpha,\beta]$ is taken for illustration. Take X and a according to domain and codomain as:

1. $X:[x,y]$ if domain is R^2 , $a:[\alpha,\beta]$ if co-domain is R^2
2. $X:[x,y,z]$ if domain is R^3 , $a:[\alpha,\beta,\gamma]$ if co-domain is R^3
3. $X:[w,x,y,z]$ if domain is R^4 , $a:[\alpha,\beta,\gamma,\delta]$ if co-domain is R^4 and so on.

Worked Examples:

Problem 1. Write a program to find the matrix of linear transformation $T: R^3 \rightarrow R^2$ defined by $T(x, y, z) = (x + y, y + z)$ with respect to standard bases.

Program:

```
T(x):=[x[1]+x[2],x[2]+x[3]]$
X:[x,y,z]$
A:[[1,0,0],[0,1,0],[0,0,1]]$
B:[[1,0],[0,1]]$
L: makelist(T(A[i]),i,1,length(A))$
M: transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$
```

```
T(x):=[x[1]+x[2],x[2]+x[3]]$
X:[x,y,z]$
A:[[1,0,0],[0,1,0],[0,0,1]]$
B:[[1,0],[0,1]]$
L: makelist(T(A[i]),i,1,length(A))$
M: transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$

Given Linear Transformation is T(x,y,z)= [y + x , z + y]
Basis of domain is B1= [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
Basis of co-domain is B2= [[1, 0], [0, 1]]
The Matrix of T w.r.t. given bases is [T]=  $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ 
```

Output:

Given Linear Transformation is $T(x, y, z) = [y + x, z + y]$

Basis of domain is $B1 = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]$

Basis of co – domain is $B2 = [[1, 0], [0, 1]]$

The Matrix of T w.r.t. given bases is $[T] = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$

Problem 2. Write a program to find the matrix of linear transformation $T: \mathbf{R}^2 \rightarrow \mathbf{R}^3$ defined by $T(x, y) = (2y - x, y, 3y - 3x)$ with respect to standard bases.

Program:

```
T(x):=[2*x[2]-x[1],x[2],3*x[2]-3*x[1]]$
X:[x,y]$
A:[[1,0],[0,1]]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(A[i]),i,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix T w.r.t. given bases is [T]=",M)$
```

Output:

Given Linear Transformation is $T(x, y) = [2y - x, y, 3y - 3x]$

Basis of domain is $B1 = [[1, 0], [0, 1]]$

Basis of co – domain is $B2 = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]$

The Matrix of T w.r.t. given bases is $[T] = \begin{pmatrix} -1 & 2 \\ 0 & 1 \\ -3 & 3 \end{pmatrix}$

```
T(x):=[2*x[2]-x[1],x[2],3*x[2]-3*x[1]]$
X:[x,y]$
A:[[1,0],[0,1]]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(A[i]),i,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix T w.r.t. given bases is [T]=",M)$

Given Linear Transformation is T(x,y)= [2 y -x ,y ,3 y -3 x ]
Basis of domain is B1= [[1, 0],[0, 1]]
Basis of co-domain is B2= [[1, 0, 0],[0, 1, 0],[0, 0, 1]]

The Matrix T w.r.t. given bases is [T]=  $\begin{pmatrix} -1 & 2 \\ 0 & 1 \\ -3 & 3 \end{pmatrix}$ 
```

Problem 3. Write a program to find the matrix of linear transformation $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (x \cos(\theta) - y \sin(\theta), x \sin(\theta) + y \cos(\theta))$ with respect to standard bases.

Program:

```
T(x):=[x[1]*cos(theta)-x[2]*sin(theta), x[1]*sin(theta)+x[2]*cos(theta)]$
X:[x,y]$
A:[[1,0],[0,1]]$
B:[[1,0],[0,1]]$
L:makelist(T(A[i]),i,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix T w.r.t. given bases is [T]=",M)$
```

```
T(x):=[x[1]*cos(theta)-x[2]*sin(theta), x[1]*sin(theta)+x[2]*cos(theta)]$
X:[x,y]$
A:[[1,0],[0,1]]$
B:[[1,0],[0,1]]$
L:makelist(T(A[i]),i,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix T w.r.t. given bases is [T]=",M)$

Given Linear Transformation is T(x,y)= [x cos(theta)-y sin(theta), x sin(theta)+y cos(theta)]
Basis of domain is B1= [[1,0],[0,1]]
Basis of co-domain is B2= [[1,0],[0,1]]

The Matrix T w.r.t. given bases is [T]=  $\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ 
```

Output:

Given Linear Transformation is $T(x, y) = [x \cos(\theta) - y \sin(\theta), x \sin(\theta) + y \cos(\theta)]$
 Basis of domain is $B1 = [[1,0], [0,1]]$
 Basis of co – domain is $B2 = [[1,0], [0,1]]$
 The Matrix of T w.r.t. given bases is $[T] = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$

Problem 4. Write a program to find the matrix of linear transformation $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x + 2y - z, y + z, x + y - 2z)$ with respect to standard bases.

Program:

```
T(x):=[x[1]+2*x[2]-x[3], x[2]+x[3], x[1]+x[2]-2*x[3]]$
X:[x,y,z]$
A:[[1,0,0],[0,1,0],[0,0,1]]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(A[i]),i,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix T w.r.t. given bases is [T]=",M)$
```

```
T(x):=[x[1]+2*x[2]-x[3], x[2]+x[3], x[1]+x[2]-2*x[3]]$
X:[x,y,z]$
A:[[1,0,0],[0,1,0],[0,0,1]]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(A[i]),i,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix T w.r.t. given bases is [T]=",M)$

Given Linear Transformation is T(x,y,z)= [-z+2 y+x, z+y, -2 z+y+x]
Basis of domain is B1= [[1,0,0],[0,1,0],[0,0,1]]
Basis of co-domain is B2= [[1,0,0],[0,1,0],[0,0,1]]

The Matrix T w.r.t. given bases is [T]=  $\begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & 1 \\ 1 & 1 & -2 \end{bmatrix}$ 
```

Output:

Given Linear Transformation is $T(x, y) = [-z + 2y + x, z + y, -2z + y + x]$

Basis of domain is $B1 = [[1,0,0], [0,1,0], [0,0,1]]$

Basis of co – domain is $B2 = [[1,0,0], [0,1,0], [0,0,1]]$

The Matrix of T w.r.t. given bases is $[T] = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 1 & 1 \\ 1 & 1 & -2 \end{pmatrix}$

Problem 5. Write a program to find the matrix of linear transformation $T: \mathbf{R}^2 \rightarrow \mathbf{R}^3$ defined by $T(x, y) = (2y - x, y, 3y - 3x)$ with respect to ordered bases of domain and co-domain $B1 = \{(1, 1), (-1, 1)\}$ and $B2 = \{(1, 1, 1), (1, -1, 1), (0, 0, 1)\}$, respectively.

Program:

```
T(x):=[2*x[2]-x[1],x[2],3*x[2]-3*x[1]]$
X:[x,y]$
A:[[1,1],[-1,1]]$
B:[[1,1,1],[1,-1,1],[0,0,1]]$
a:[α,β,γ]$
K:makelist(linsolve(B.a-T(A[i]),a),i,1,length(A))$
L:makelist(makelist(assoc(a[i],K[j]),i,1,length(a)),j,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$
```

```
T(x):=[2*x[2]-x[1],x[2],3*x[2]-3*x[1]]$
X:[x,y]$
A:[[1,1],[-1,1]]$
B:[[1,1,1],[1,-1,1],[0,0,1]]$
a:[α,β,γ]$
K:makelist(linsolve(B.a-T(A[i]),a),i,1,length(A))$
L:makelist(makelist(assoc(a[i],K[j]),i,1,length(a)),j,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$

Given Linear Transformation is T(x,y)= [2 y - x , y , 3 y - 3 x]
Basis of domain is B1= [[1, 1], [-1, 1]]
Basis of co-domain is B2= [[1, 1, 1], [1, -1, 1], [0, 0, 1]]

The Matrix of T w.r.t. given bases is [T]=  $\begin{pmatrix} 1 & 2 \\ 0 & 1 \\ -1 & 3 \end{pmatrix}$ 
```

Output:

Given Linear Transformation is $T(x, y) = [2y - x, y, 3y - 3x]$

Basis of domain is $B1 = [[1,1], [-1,1]]$

Basis of co – domain is $B2 = [[1,1,1], [1, -1,1], [0,0,1]]$

The Matrix of T w.r.t. given bases is $[T] = \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ -1 & 3 \end{pmatrix}$

Problem 6. Write a program to find the matrix of linear transformation $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (x + 4y, 2x - 3y)$ with respect to ordered bases of domain and co-domain $B1 = \{(1, 0), (0, 1)\}$ and $B2 = \{(1, 3), (2, 5)\}$, respectively.

Program:

```
T(x):=[x[1]+4*x[2],2*x[1]-3*x[2]]$
X:[x,y]$
A:[[1,0],[0,1]]$
B:[[1,3],[2,5]]$
a:[α,β]$
K:makelist(linsolve(B.a-T(A[i]),a),i,1,length(A))$
L:makelist(makelist(assoc(a[i],K[j]),i,1,length(a)),j,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$
```

Output:

Given Linear Transformation is $T(x, y) = [4y + x, 2x - 3y]$

Basis of domain is $B1 = [[1, 0], [0, 1]]$

Basis of co – domain is $B2 = [[1, 3], [2, 5]]$

The Matrix of T w.r.t. given bases is $[T] = \begin{pmatrix} -1 & -26 \\ 1 & 15 \end{pmatrix}$

```
T(x):=[x[1]+4*x[2],2*x[1]-3*x[2]]$
X:[x,y]$
A:[[1,0],[0,1]]$
B:[[1,3],[2,5]]$
a:[α,β]$
K:makelist(linsolve(B.a-T(A[i]),a),i,1,length(A))$
L:makelist(makelist(assoc(a[i],K[j]),i,1,length(a)),j,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$

Given Linear Transformation is T(x,y)= [4 y +x, 2 x -3 y]
Basis of domain is B1= [[1, 0], [0, 1]]
Basis of co-domain is B2= [[1, 3], [2, 5]]

The Matrix of T w.r.t. given bases is [T]=  $\begin{pmatrix} -1 & -26 \\ 1 & 15 \end{pmatrix}$ 
```

Problem 7. Write a program to find the matrix of linear transformation $T: \mathbf{R}^4 \rightarrow \mathbf{R}^3$ defined by

$T(\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z}) = (\mathbf{w} + \mathbf{x} + 2\mathbf{y} + 3\mathbf{z}, \mathbf{w} + \mathbf{x} - \mathbf{z}, \mathbf{w} + 2\mathbf{x})$ with respect to ordered bases of domain and co-domain $B1 = \{(\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{2}), (\mathbf{1}, -\mathbf{1}, \mathbf{0}, \mathbf{0}), (\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}), (\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0})\}$ and $B2 = \{(\mathbf{1}, \mathbf{2}, \mathbf{3}), (\mathbf{1}, -\mathbf{1}, \mathbf{1}), (\mathbf{2}, \mathbf{1}, \mathbf{1})\}$, respectively.

Program:

```
T(x):=[x[1]+x[2]+2*x[3]+3*x[4],x[1]+x[2]-x[4],x[1]+2*x[2]]$
X:[w,x,y,z]$
A:[[1,1,1,2],[1,-1,0,0],[0,0,1,1],[0,1,0,0]]$
B:[[1,2,3],[1,-1,1],[2,1,1]]$
a:[α,β,γ]$
K:makelist(linsolve(B.a-T(A[i]),a),i,1,length(A))$
L:makelist(makelist(assoc(a[i],K[j]),i,1,length(a)),j,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$
```

Output:

Given Linear Transformation is $T(w, x, y, z) = [3z + 2y + x + w, -z + x + w, 2x + w]$

Basis of domain is $B1 = [[1, 1, 1, 2], [1, -1, 0, 0], [0, 0, 1, 1], [0, 1, 0, 0]]$

Basis of co – domain is $B2 = [[1, 2, 3], [1, -1, 1], [2, 1, 1]]$

The Matrix of T w.r.t. given bases is $[T] = \begin{pmatrix} -\frac{11}{9} & -\frac{1}{3} & -\frac{11}{9} & \frac{5}{9} \\ \frac{19}{9} & -\frac{1}{3} & \frac{10}{9} & \frac{2}{9} \\ \frac{41}{9} & \frac{1}{3} & \frac{23}{9} & \frac{1}{9} \end{pmatrix}$

```
T(x):=[x[1]+x[2]+2*x[3]+3*x[4],x[1]+x[2]-x[4],x[1]+2*x[2]]$
X:[w,x,y,z]$
A:[[1,1,1,2],[1,-1,0,0],[0,0,1,1],[0,1,0,0]]$
B:[[1,2,3],[1,-1,1],[2,1,1]]$
a:[α,β,γ]$
K:makelist(linsolve(B.a-T(A[i]),a),i,1,length(A))$
L:makelist(makelist(assoc(a[i],K[j]),i,1,length(a)),j,1,length(A))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Basis of domain is B1=",A)$
print("Basis of co-domain is B2=",B)$
print("The Matrix of T w.r.t. given bases is [T]=",M)$

Given Linear Transformation is T(w,x,y,z)= [3 z+2 y+x+w, -z+x+w, 2 x+w]
Basis of domain is B1= [[1, 1, 1, 2], [1, -1, 0, 0], [0, 0, 1, 1], [0, 1, 0, 0]]
Basis of co-domain is B2= [[1, 2, 3], [1, -1, 1], [2, 1, 1]]

The Matrix of T w.r.t. given bases is [T]=
```

$$\begin{pmatrix} -\frac{11}{9} & -\frac{1}{3} & -\frac{11}{9} & \frac{5}{9} \\ \frac{19}{9} & -\frac{1}{3} & \frac{10}{9} & \frac{2}{9} \\ \frac{41}{9} & \frac{1}{3} & \frac{23}{9} & \frac{1}{9} \end{pmatrix}$$

Exercise:

I. Write a program to find the matrix of given linear transformation with respect to standard bases.

1. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (x, -y)$ (Answer: $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$)

2. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (y, -x, z)$ (Answer: $\begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$)

3. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined by $T(x, y) = (-x + 2y, y, -3x + 3y)$ (Answer: $\begin{pmatrix} -1 & 2 \\ 0 & 1 \\ -3 & 3 \end{pmatrix}$)

4. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (z - 2y, x + 2y - z)$ (Answer: $\begin{pmatrix} 0 & -2 & 1 \\ 1 & 2 & -1 \end{pmatrix}$)

II. Write a program to find the matrix of given linear transformation with respect to given bases.

1. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (x, -y)$ with respect to bases

$B1 = \{(1, 1), (1, 0)\}$ and $B2 = \{(2, 3), (4, 5)\}$

(Answer: $\begin{pmatrix} -\frac{9}{2} & -\frac{5}{2} \\ \frac{5}{2} & \frac{3}{2} \end{pmatrix}$)

2. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (y, -x, z)$ with respect to bases

$B1 = B2 = \{(1, 1, 0), (0, 1, 1), (1, 0, 1)\}$

(Answer: $\begin{pmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$)

3. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined by $T(x, y) = (-x + 2y, y, -3x + 3y)$ with respect to bases

$B1 = \{(1, 2), (-2, 1)\}$ and $B2 = \{(-1, 0, 2), (1, 2, 3), (1, -1, -1)\}$ (Answer: $\begin{pmatrix} -\frac{2}{7} & \frac{19}{7} \\ \frac{11}{7} & \frac{18}{7} \\ \frac{8}{7} & \frac{29}{7} \end{pmatrix}$)

4. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (z - 2y, x + 2y - z)$ with respect to bases

$B1 = \{(1, 2, 3), (1, -1, 1), (2, 1, 1)\}$ and $B2 = \{(2, 1), (-3, 4)\}$ (Answer: $\begin{pmatrix} \frac{2}{11} & \frac{6}{11} & \frac{5}{11} \\ \frac{5}{11} & -\frac{7}{11} & \frac{7}{11} \end{pmatrix}$)

Program 6

Program to find the Eigenvalues and Eigenvectors of a given linear transformation.

Aim: To find the Eigenvalues and Eigenvectors of a given linear transformation / matrix using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
. (dot)	The operator . represents noncommutative multiplication and scalar product. It is used for usual multiplication of matrices.
* (asterisk)	The operator * represents commutative multiplication
^	Exponentiation operator or power or index
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
L[i]	Returns i-th element of the list L
'	The single quote operator ' prevents evaluation.
makelist (expr, i, i ₀ , i _{max})	Returns the list of elements obtained when ev (expr, i=j) is applied to the elements j of the sequence: i ₀ , i ₀ + 1, i ₀ + 2, ..., with j less than or equal to i _{max} .
length (expr)	Returns the number of parts in the external (displayed) form of expr
:=	The function definition operator
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print ("text", expr)\$	Displays text within inverted commas and evaluates and displays expr
transpose (M)	Returns the transpose of M.
apply('matrix,L)	Converting nested lists L to matrix
reverse (list)	Reverses the order of the members of the list (not the members themselves)
table_form()	Displays a 2D list in a form that is more readable than the output from Maxima's default output routine. The input is a list of one or more lists.
push (item, list)	push prepends the item item to the list list and returns a copy of the new list.
eigenvalues (M) or eivals (M)	Returns a list of two lists containing the eigenvalues of the matrix M. The first sublist of the return value is the list of eigenvalues of the matrix, and the second sublist is the list of the multiplicities of the eigenvalues in the corresponding order.

eigenvectors (M) or eivects (M)	Computes eigenvectors of the matrix M . The return value is a list of two elements. The first is a list of the eigenvalues of M and a list of the multiplicities of the eigenvalues. The second is a list of lists of eigenvectors. There is one list of eigenvectors for each eigenvalue. There may be one or more eigenvectors in each list.
---	--

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Eigen value and Eigen vector of a Linear transformation: Let V be a vector space over a field F and $T: V \rightarrow V$ be a linear transformation. A scalar $\lambda \in F$ is called an eigen value / characteristic value of T if $T(v) = \lambda v$ for some non-zero vector $v \in V$. Then v is called an eigen vector / characteristic vector corresponding to the eigen value λ .

Similarly, a non-zero vector $v \in V$ is called an eigen vector / characteristic vector of T if $T(v) = \lambda v$ for some scalar $\lambda \in F$. Then $\lambda \in F$ is called an eigen value / characteristic value corresponding to the eigen vector v .

Properties of eigen values and eigen vectors:

1. The number of distinct eigen values of a linear transformation of a vector space of dimension n is at most n .
2. An eigen value of a linear transformation may be repeated. The number of times an eigen value is repeated is called its **algebraic multiplicity**.
3. A non-repeated eigen value can have only one linearly independent eigen vector.
4. A repeated eigen value can have one or more linearly independent eigen vectors. The number of linearly independent eigen vectors of an eigen value is called its **geometric multiplicity**. For any eigen value: **geometric multiplicity** $\leq$ **algebraic multiplicity**.
5. Geometric multiplicity of an eigen value is the dimension of its eigen space.
6. An eigen value is called defective if its **geometric multiplicity** $<$ **algebraic multiplicity**
7. A linear transformation is diagonalizable if all its eigen values are non-defective. In other words, all eigen values have their geometric multiplicity = their algebraic multiplicity. In this case corresponding eigen vectors form a basis of the vector space.
8. $0 \in F$, scalar zero can be an eigen value, but $\mathbf{0} \in V$, zero vector can **never** be an eigen vector.

Program:

Program to find the eigen values and eigen vectors of a linear transformation $T: R^n \rightarrow R^n$

```
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
B:Standard basis of given vector space$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]+$
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]+$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
table_form(reverse(N))$
```

Note: 1. Here X:[x,y,z] is taken for illustration purpose.

Take X:[x,y] if domain is R^2 , X:[x,y,z] if domain is R^3 ,
X:[w,x,y,z] if domain is R^4 and so on.

2. Take standard basis as: B:[1,0],[0,1] for R^2 , B:[1,0,0],[0,1,0],[0,0,1] for R^3 ,
B:[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1] for R^4 and so on.

Program to find the eigen values and eigen vectors of a matrix

```
M:matrix([R1],[R2],...[Rn])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]+$
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]+$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Matrix is M=",M)$
table_form(reverse(N))$
```

Worked Examples:

Problem 1. Write a program to find the eigen values and eigen vectors of the linear transformation

$$T: \mathbb{R}^2 \rightarrow \mathbb{R}^2 \text{ defined by } T(x, y) = (4y + x, 3y + 2x)$$

Program:

```
T(x):=[x[1]+4*x[2],2*x[1]+3*x[2]]$
```

```
X:[x,y]$
```

```
B:[[1,0],[0,1]]$
```

```
L:makelist(T(B[i]),i,1,length(B))$
```

```
M:transpose(apply('matrix,L))$
```

```
[vals, vecs]:eigenvectors(M)$
```

```
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]
```

```
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]
```

```
for i thru length(vals[1]) do N:push(N(i),N)$
```

```
print("Given Linear Transformation is T(x,y)=",T(X))$
```

```
table_form(reverse(N))$
```

```
T(x):=[x[1]+4*x[2],2*x[1]+3*x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]
```

Given Linear Transformation is T(x,y)= [4 y + x , 3 y + 2 x]

Eigen Value Multiplicity Linearly Independent Eigen vectors

$\lambda_1 = 5$ 1 { [1, 1] }

$\lambda_2 = -1$ 1 { [1, - $\frac{1}{2}$] }

Output:

Given Linear Transformation is T(x,y) = [4y + x, 3y + 2x]

Eigen Value Multiplicity Linearly Independent Eigen vectors

$\lambda_1 = 5$ 1 { [1, 1] }

$\lambda_2 = -1$ 1 { [1, - $\frac{1}{2}$] }

Problem 2. Write a program to find the eigen values and eigen vectors of the linear transformation

$$T: \mathbb{R}^2 \rightarrow \mathbb{R}^2 \text{ defined by } T(x, y) = (y, 0)$$

Program:

```
T(x):=[x[2],0]$
```

```
X:[x,y]$
```

```
B:[[1,0],[0,1]]$
```

```
L:makelist(T(B[i]),i,1,length(B))$
```

```
M:transpose(apply('matrix,L))$
```

```
[vals, vecs]:eigenvectors(M)$
```

```
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]
```

```
T(x):=[x[2],0]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]
```

Given Linear Transformation is T(x,y)= [y , 0]

Eigen Value Multiplicity Linearly Independent Eigen vectors

$\lambda_1 = 0$ 2 { [1, 0] }

```

N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i]))$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y)=",T(X))$
table_form(reverse(N))$

```

Output:

Given Linear Transformation is $T(x, y) = [y, 0]$

Eigen Value	Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 0$	2	$\{[1, 0]\}$

Problem 3. Write a program to find the eigen values and eigen vectors of the linear transformation

$T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (2x + 2y + z, x + 3y + z, x + 2y + 2z)$

Program:

```

T(x):=[2*x[1]+2*x[2]+x[3],x[1]+3*x[2]+x[3],x[1]+2*x[2]+2*x[3]]$
X:[x,y,z]$

```

```

B:[[1,0,0],[0,1,0],[0,0,1]]$

```

```

L:makelist(T(B[i]),i,1,length(B))$

```

```

M:transpose(apply('matrix,L))$

```

```

[vals, vecs]:eigenvectors(M)$

```

```

N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]+$

```

```

N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i]))$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
table_form(reverse(N))$

```

```

T(x):=[2*x[1]+2*x[2]+x[3],x[1]+3*x[2]+x[3],x[1]+2*x[2]+2*x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]+$
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i]))$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
table_form(reverse(N))$

Given Linear Transformation is T(x,y,z)= [z+2 y+2 x , z+3 y+x , 2 z+2 y+x]

Eigen Value  Multiplicity  Linearly Independent Eigen vectors

λ1=5          1           {[1,1,1]}

λ2=1          2           {[0,1,-2],[1,0,-1]}

```

Output:

Given Linear Transformation is $T(x, y, z) = [z + 2y + 2x, z + 3y + x, 2z + 2y + x]$

Eigen Value	Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 5$	1	$\{[1, 1, 1]\}$
$\lambda_2 = 1$	2	$\{[0, 1, -2], [1, 0, -1]\}$

Problem 4. Write a program to find the eigen values and eigen vectors of the linear transformation

$$T: \mathbf{R}^4 \rightarrow \mathbf{R}^4 \text{ defined by } T(w, x, y, z) = (0, x, y, z)$$

Program:

```
T(x):=[0,x[2],x[3],x[4]]$
```

```
X:[w,x,y,z]$
```

```
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
```

```
L:makelist(T(B[i]),i,1,length(B))$
```

```
M:transpose(apply('matrix,L))$
```

```
[vals, vecs]:eigenvalues(M)$
```

```
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]]$
```

```
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]$
```

```
for i thru length(vals[1]) do N:push(N(i),N)$
```

```
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
```

```
table_form(reverse(N))$
```

```
T(x):=[0,x[2],x[3],x[4]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvalues(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
table_form(reverse(N))$

Given Linear Transformation is T(w,x,y,z)= [0,x,y,z]

Eigen Value  Multiplicity  Linearly Independent Eigen vectors
λ1=0         1           {[1,0,0,0]}
λ2=1         3           {[0,0,0,1],[0,0,1,0],[0,1,0,0]}
```

Output:

Given Linear Transformation is $T(w, x, y, z) = [0, x, y, z]$

Eigen Value Multiplicity Linearly Independent Eigen vectors

$\lambda_1 = 0$ 1 {[1,0,0,0]}

$\lambda_2 = 1$ 3 {[0,0,0,1], [0,0,1,0], [0,1,0,0]}

Problem 5. Write a program to find the eigen values and eigen vectors of the matrix $M = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$

Program:

```
M:matrix([1,2,3],[2,1,0],[-1,0,1])$
```

```
[vals, vecs]:eigenvalues(M)$
```

```
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]]$
```

```
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]$
```

```
for i thru length(vals[1]) do N:push(N(i),N)$
```

```
print("Given Matrix is M=",M)$
```

```
table_form(reverse(N))$
```

```
M:matrix([1,2,3],[2,1,0],[-1,0,1])$
[vals, vecs]:eigenvalues(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Matrix is M=",M)$
table_form(reverse(N))$

Given Matrix is M=

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$


Eigen Value  Multiplicity  Linearly Independent Eigen vectors
λ1=1         1           {[0,1,- $\frac{2}{3}$ ]}
λ2=2         1           {[1,2,-1]}
λ3=0         1           {[1,-2,1]}
```

Output:

$$\text{Given Matrix is } M = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Eigen Value	Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 1$	1	$\left\{ \begin{bmatrix} 0 \\ 1 \\ -\frac{2}{3} \end{bmatrix} \right\}$
$\lambda_2 = 2$	1	$\{[1, 2, -1]\}$
$\lambda_3 = 0$	1	$\{[1, -2, 1]\}$

Problem 6. Write a program to find the eigen values and eigen vectors of the matrix $M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Program:

```
M:matrix([0,1,0,0],[0,0,2,0],[0,0,0,3],[0,0,0,0])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]+$
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]+$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Matrix is M=",M)$
table_form(reverse(N))$
```

Output:

$$\text{Given Matrix is } M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Eigen Value	Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 0$	4	$\{[1, 0, 0, 0]\}$

```
M:matrix([0,1,0,0],[0,0,2,0],[0,0,0,3],[0,0,0,0])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","Multiplicity","Linearly Independent Eigen vectors"]]+$
N(i):=[λ[i]=vals[1][i],vals[2][i],setify(vecs[i])]+$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Matrix is M=",M)$
table_form(reverse(N))$
```

$$\text{Given Matrix is } M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Eigen Value	Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 0$	4	$\{[1, 0, 0, 0]\}$

Exercise:

I. Write a program to find the eigen values and eigen vectors of the given linear transformation

1. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (3x + y, 6x + 2y)$ (Answer: $\lambda_1 = 0, \lambda_2 = 5$)

2. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (x, 2x + y, 3x + 2y)$ (Answer: $\lambda_1 = 0, \lambda_2 = 1$ (twice repeated))

3. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (5x - 6y - 6z, -x + 4y + 2z, 3x - 6y - 4z)$

(Answer: $\lambda_1 = 1, \lambda_2 = 2$ (twice repeated))

4. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (x - y + 2z, y, x + 2y + z)$

(Answer: $\lambda_1 = 1, \lambda_2 = 1 + \sqrt{2}, \lambda_3 = 1 - \sqrt{2}$)

5. $T: \mathbb{R}^4 \rightarrow \mathbb{R}^4$ defined by $T(w, x, y, z) = (0, w, x, y)$ (Answer: $\lambda_1 = 0$ (repeated 4 times))

II. Write a program to find the eigen values and eigen vectors of the given matrix

1. $M = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}$ (Answer: $\lambda_1 = 1, \lambda_2 = 3$)

2. $M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$ (Answer: $\lambda_1 = 1, \lambda_2 = 2, \lambda_3 = 3$)

3. $M = \begin{pmatrix} 0 & 0 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & -2 \end{pmatrix}$ (Answer: $\lambda_1 = -2, \lambda_2 = -1, \lambda_3 = 1$)

4. $M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (Answer: $\lambda_1 = -1, \lambda_2 = 1$ (twice repeated))

5. $M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ -2 & -2 & 2 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix}$ (Answer: $\lambda_1 = 1$ (twice repeated), $\lambda_2 = 0$ (twice repeated))

Program 7

Program on Rank – Nullity Theorem.

Aim: To verify Rank – Nullity theorem for given linear transformation / matrix using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
. (dot)	The operator . represents noncommutative multiplication and scalar product. It is used for usual multiplication of matrices.
* (asterisk)	The operator * represents commutative multiplication
^	Exponentiation operator or power or index
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
L[i]	Returns i-th element of the list L
'	The single quote operator ' prevents evaluation.
and	The logical conjunction operator
length (expr)	Returns the number of parts in the external (displayed) form of <i>expr</i>
:=	The function definition operator
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print (“text”, expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
transpose (M)	Returns the transpose of <i>M</i> .
apply('matrix,L)	Converting nested lists L to matrix
matrix_size (M)	Return a two-member list that gives the number of rows and columns, respectively of the matrix <i>M</i> .
columnspace (M)	If <i>M</i> is a matrix, return span (v ₁ , ..., v _n), where the set {v ₁ , ..., v _n } is a basis for the column space of <i>M</i> .
rank (M)	Computes the rank of the matrix <i>M</i> .
nullspace (M)	If <i>M</i> is a matrix, return span (v ₁ , ..., v _n), where the set {v ₁ , ..., v _n } is a basis for the nullspace of <i>M</i>
nullity (M)	If <i>M</i> is a matrix, return the dimension of the nullspace of <i>M</i> .

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Range / Range space and rank of a Linear transformation: Let U and V be vector spaces over a field F and $T: U \rightarrow V$ be a linear transformation. The range of T is the set $R(T) = \{T(u) \in V \mid u \in U\}$. Clearly, the range or range space $R(T)$ of T is a subspace of codomain V and the dimension of $R(T)$ is called the rank of T and is denoted by $rank(T)$ or $r(T)$.

Kernel / Null space and nullity of a Linear transformation: Let U and V be vector spaces over a field F and $T: U \rightarrow V$ be a linear transformation. The kernel / null space of T is the set $N(T) = \{u \in U \mid T(u) = 0 \in V\}$. Clearly, the kernel or null space $N(T)$ of T is a subspace of domain U and the dimension of $N(T)$ is called the nullity of T and is denoted by $nullity(T)$ or $n(T)$.

Rank - Nullity Theorem for a linear transformation: Let U and V be vector spaces over a field F and $T: U \rightarrow V$ be a linear transformation. Suppose U is finite dimensional of dimension n . Then,

$$rank(T) + nullity(T) = dim(U) \quad \text{i.e.,} \quad r(T) + n(T) = n$$

i.e., the sum of rank and nullity of a linear transformation is equal to the dimension of its domain. It is also called Rank-Nullity-Dimension theorem.

Rank - Nullity Theorem for a matrix: Let A be an $m \times n$ matrix. Then the number of columns of A is the sum of the rank of A and the nullity of A . That is,

$$rank(A) + nullity(A) = \text{Number of Columns} \quad \text{i.e.,} \quad r(T) + n(T) = n$$

Program:

Program to verify rank-nullity theorem for a matrix M

```
M:matrix([R1],[R2],[R3],...[Rm])$
n:matrix_size (M)[2]$
print("Given Matrix is M=",M)$
print("Range Space of M=",columnspace(M))$
print("Null Space of M=",nullspace(M))$
print("Rank of M=",rank(M))$
print("Nullity of M=",nullity(M))$
print("r(M)+n(M)=",rank(M)+nullity(M))$
print("Number of columns of M=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Number of columns") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Number of columns") and
print("Rank - Nullity theorem is not verified")$
```

Program to verify rank-nullity theorem for a linear transformation $T: U \rightarrow V$

```
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
B: Standard ordered basis of domain$
L: makelist(T(B[i]),i,1,length(B))$
M: transpose(apply('matrix,L))$
n: matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity  $\neq$  Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$
```

Note: 1. In the above program $X:[x,y,z]$ is taken for illustration.

Take $X:[x,y]$ if $U = R^2$, $X:[x,y,z]$ if $U = R^3$, $X:[w,x,y,z]$ if $U = R^4$ and so on.

2. Take standard bases as:

$[[1,0],[0,1]]$ for R^2

$[[1,0,0],[0,1,0],[0,0,1]]$ for R^3

$[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$ for R^4 and so on.

Worked Examples:

Problem 1. Write a program to verify rank-nullity theorem for a linear transformation $T: \mathbf{R}^3 \rightarrow \mathbf{R}^2$ defined by $T(x, y, z) = (x + y, y + z)$

Program:

```
T(x):=[x[1]+x[2], x[2]+x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$
```

```
) T(x):=[x[1]+x[2], x[2]+x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$
Given Linear Transformation is T(x,y,z)= [y + x , z + y]
Range Space of T= span  $\left( \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right)$ 
Null Space of T= span  $\left( \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} \right)$ 
Rank of T= 2
Nullity of T= 1
r(T)+n(T)= 3
Dimension of Domain= 3
Rank + Nullity = Dimension of domain
Rank - Nullity theorem is verified
```

Output:

Given Linear Transformation is $T(x, y, z) = [y + x, z + y]$

Range Space of $T = \text{span} \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right)$

Null Space of $T = \text{span} \left(\begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} \right)$

Rank of $T = 2$

Nullity of $T = 1$

$r(T) + n(T) = 3$

Dimension of Domain = 3

Rank + Nullity = Dimension of domain

Rank - Nullity theorem is verified

Problem 2. Write a program to verify rank-nullity theorem for a linear transformation $T: \mathbf{R}^3 \rightarrow \mathbf{R}^4$ defined by $T(x, y, z) = (0, x + y + z, y - z, 2x + 4z)$

Program:

```
T(x):=[0,x[1]+x[2]+x[3], x[2]-x[3], 2*x[1]+4*x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$
```

```
T(x):=[0,x[1]+x[2]+x[3], x[2]-x[3], 2*x[1]+4*x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$

Given Linear Transformation is T(x,y,z)= [0, z+y+x, y-z, 4z+2x]

Range Space of T= span  $\left( \begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \right)$ 

Null Space of T= span  $\left( \begin{pmatrix} -2 \\ 1 \\ 1 \end{pmatrix} \right)$ 

Rank of T= 2
Nullity of T= 1
r(T)+n(T)= 3
Dimension of Domain= 3
Rank + Nullity = Dimension of domain
Rank - Nullity theorem is verified
```

Output:

Given Linear Transformation is $T(x, y, z) = [0, z + y + x, y - z, 4z + 2x]$

$$\text{Range Space of } T = \text{span} \left(\begin{pmatrix} 0 \\ 1 \\ 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \right)$$

$$\text{Null Space of } T = \text{span} \left(\begin{pmatrix} -2 \\ 1 \\ 1 \end{pmatrix} \right)$$

Rank of $T = 2$

Nullity of $T = 1$

$r(T) + n(T) = 3$

Dimension of Domain = 3

Rank + Nullity = Dimension of domain

Rank - Nullity theorem is verified

Problem 3. Write a program to verify rank-nullity theorem for a linear transformation $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x + y, x - y, 2x + z)$

Program:

```
T(x):=[0,x[1]+x[2]+x[3], x[2]-x[3], 2*x[1]+4*x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$
```

```
T(x):=[x[1]+x[2], x[1]-x[2], 2*x[1]+x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$
Given Linear Transformation is T(x,y,z)= [y + x , x - y , z + 2 x]

Range Space of T= span  $\left( \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \right)$ 

Null Space of T= span(?)
Rank of T= 3
Nullity of T= 0
r(T)+n(T)= 3
Dimension of Domain= 3
Rank + Nullity = Dimension of domain
```

Output:

Given Linear Transformation is $T(x, y, z) = [y + x, x - y, z + 2x]$

Range Space of $T = \text{span} \left(\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \right)$

Null Space of $T = \text{span}(?)$

Rank of $T = 3$

Nullity of $T = 0$

$r(T) + n(T) = 3$

Dimension of Domain = 3

Rank + Nullity = Dimension of domain

Rank - Nullity theorem is verified

Note: $\text{span}(?)$ means $\text{span}(\)$ i.e., span of the empty set and $\text{span}(?) = \text{span}(\phi) = \{0\}$

Problem 4. Write a program to verify rank-nullity theorem for a linear transformation $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined by $T(x, y) = (-x + 2y, y, -3x + 3y)$

Program:

```
T(x):=[-x[1]+2*x[2],x[2],-3*x[1]+3*x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$
```

```
T(x):=[-x[1]+2*x[2],x[2],-3*x[1]+3*x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
n:matrix_size (M)[2]$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Range Space of T=",columnspace(M))$
print("Null Space of T=",nullspace(M))$
print("Rank of T=",rank(M))$
print("Nullity of T=",nullity(M))$
print("r(T)+n(T)=",rank(M)+nullity(M))$
print("Dimension of Domain=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Dimension of domain") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Dimension of Domain") and
print("Rank - Nullity theorem is not verified")$

Given Linear Transformation is T(x,y)= [2 y - x , y, 3 y - 3 x ]

Range Space of T= span  $\left( \begin{pmatrix} -1 \\ 0 \\ -3 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} \right)$ 

Null Space of T= span(?)
Rank of T= 2
Nullity of T= 0
r(T)+n(T)= 2
Dimension of Domain= 2
Rank + Nullity = Dimension of domain
Rank - Nullity theorem is verified
```

Output:

Given Linear Transformation is $T(x, y, z) = [2y - x, y, 3y - 3x]$

Range Space of $T = \text{span} \left(\begin{pmatrix} -1 \\ 0 \\ -3 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} \right)$

Null Space of $T = \text{span}(?)$

Rank of $T = 2$

Nullity of $T = 0$

$r(T) + n(T) = 2$

Dimension of Domain = 2

Rank + Nullity = Dimension of domain

Rank - Nullity theorem is verified

Note: $\text{span}(?)$ means $\text{span}(\)$ i.e., span of the empty set and $\text{span}(?) = \text{span}(\phi) = \{0\}$

Problem 5. Write a program to verify rank-nullity theorem for the matrix $M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ -2 & -2 & 2 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix}$

Program:

```
M:matrix([1,1,0,0],[-1,-1,0,0],[-2,-2,2,1],[1,1,-1,0])$
n:matrix_size (M)[2]$
print("Given Matrix is M=",M)$
print("Range Space of M=",columnspace(M))$
print("Null Space of M=",nullspace(M))$
print("Rank of M=",rank(M))$
print("Nullity of M=",nullity(M))$
print("r(M)+n(M)=",rank(M)+nullity(M))$
print("Number of columns of M=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Number of columns") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Number of columns") and
print("Rank - Nullity theorem is not verified")$
```

Output:

$$\text{Given Matrix is } M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ -2 & -2 & 2 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix}$$

$$\text{Range Space of } M = \text{span} \left(\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 2 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -2 \\ 1 \end{pmatrix} \right)$$

$$\text{Null Space of } M = \text{span} \left(\begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix} \right)$$

Rank of $M = 3$
 Nullity of $M = 1$
 $r(M) + n(M) = 4$
 Number of columns of $M = 4$
 Rank + Nullity = Number of columns
 Rank - Nullity theorem is verified

```
M:matrix([1,1,0,0],[-1,-1,0,0],[-2,-2,2,1],[1,1,-1,0])$
n:matrix_size (M)[2]$
print("Given Matrix is M=",M)$
print("Range Space of M=",columnspace(M))$
print("Null Space of M=",nullspace(M))$
print("Rank of M=",rank(M))$
print("Nullity of M=",nullity(M))$
print("r(M)+n(M)=",rank(M)+nullity(M))$
print("Number of columns of M=",n)$
if rank(M)+nullity(M)=n then
print("Rank + Nullity = Number of columns") and
print("Rank - Nullity theorem is verified") else
print("Rank + Nullity ≠ Number of columns") and
print("Rank - Nullity theorem is not verified")$
```

Given Matrix is $M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ -2 & -2 & 2 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix}$

Range Space of $M = \text{span} \left(\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 2 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -2 \\ 1 \end{pmatrix} \right)$

Null Space of $M = \text{span} \left(\begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix} \right)$

Rank of $M = 3$
 Nullity of $M = 1$
 $r(M) + n(M) = 4$
 Number of columns of $M = 4$
 Rank + Nullity = Number of columns
 Rank - Nullity theorem is verified

Exercise:

I. Write a program to verify rank-nullity theorem for the following linear transformations

1. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (3x + y, 6x + 2y)$ (Ans: Rank-Nullity Theorem is verified)

2. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined by $T(x, y) = (x, x + y, y)$ (Ans: Rank-Nullity Theorem is verified)

3. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (3x - 2y + z, x - 3y - 2z)$
(Ans: Rank-Nullity Theorem is verified)

4. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^4$ defined by $T(x, y, z) = (x, x + y, x + y + z, z)$
(Ans: Rank-Nullity Theorem is verified)

5. $T: \mathbb{R}^4 \rightarrow \mathbb{R}^4$ defined by $T(w, x, y, z) = (0, w, x, y)$ (Ans: Rank-Nullity Theorem is verified)

II. Write a program to verify rank-nullity theorem for the following matrices

1. $M = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}$ (Ans: Rank-Nullity Theorem is verified)

2. $M = \begin{pmatrix} 0 & 1 & 0 & 4 & 5 \\ 0 & 1 & 0 & 4 & 5 \end{pmatrix}$ (Ans: Rank-Nullity Theorem is verified)

3. $M = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 4 & 4 \\ 5 & 5 \end{pmatrix}$ (Ans: Rank-Nullity Theorem is verified)

4. $M = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ (Ans: Rank-Nullity Theorem is verified)

5. $M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ (Ans: Rank-Nullity Theorem is verified)

Program 8

Program to verify if the given linear transformation is singular/non-singular.

Aim: To verify if the given linear transformation is singular / non-singular using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
* (asterisk)	The operator * represents commutative multiplication
^	Exponentiation operator or power or index
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
L[i]	Returns i-th element of the list L
'	The single quote operator ' prevents evaluation.
and	The logical conjunction operator
length (expr)	Returns the number of parts in the external (displayed) form of <i>expr</i>
:=	The function definition operator
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print ("text", expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
transpose (M)	Returns the transpose of <i>M</i> .
apply('matrix,L)	Converting nested lists L to matrix
columnspace (M)	If <i>M</i> is a matrix, return span (v ₁ , ..., v _n), where the set {v ₁ , ..., v _n } is a basis for the column space of <i>M</i> .
rank (M)	Computes the rank of the matrix <i>M</i> .
nullspace (M)	If <i>M</i> is a matrix, return span (v ₁ , ..., v _n), where the set {v ₁ , ..., v _n } is a basis for the nullspace of <i>M</i>
nullity (M)	If <i>M</i> is a matrix, return the dimension of the nullspace of <i>M</i> .

- Note:1. Press Shift+Enter for evaluation of commands and display of output.
2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
3. Replace dollar (\$) by semicolon (;) to see output of any input line.
4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Non-Singular Linear Transformation: Let U and V be vector spaces over a field F and $T: U \rightarrow V$ be a linear transformation. Then T is called a non-singular transformation if $T(u) = 0 \in V \Rightarrow u = 0 \in U$. In other words, T is non-singular if it maps no non-zero vector of domain to zero vector of codomain. The only vector that is mapped to zero vector of codomain is the zero vector of domain. Clearly, T is non-singular if and only if *null space of $T = N(T) = \{0\}$* and *nullity(T) = 0*. As a result, Non-singular transformation is one-one.

Singular Linear Transformation: Let U and V be vector spaces over a field F and $T: U \rightarrow V$ be a linear transformation. Then T is called a singular transformation if $\exists u \neq 0 \in U$ such that $T(u) = 0 \in V$. In other words, T is singular if it maps at least one non-zero vector of domain to the zero vector of codomain. Clearly, T is singular if and only if *null space of $T = N(T) \neq \{0\}$* and *nullity(T) ≥ 1* .

Program:

Program to verify if the given linear transformation is singular / non-singular.

```
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
B:Standard ordered basis of domain$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
```

Note: 1. In the above program $X:[x,y,z]$ is taken for illustration.

Take $X:[x,y]$ if $U = R^2$, $X:[x,y,z]$ if $U = R^3$, $X:[w,x,y,z]$ if $U = R^4$ and so on.

2. Take standard basis as: $[[1,0],[0,1]]$ for R^2 , $[[1,0,0],[0,1,0],[0,0,1]]$ for R^3

$[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$ for R^4 and so on.

Worked Examples:

Problem 1. Write a program to verify whether the linear transformation $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined by $T(x, y) = (x + y, x - y, y)$ is singular / non-singular

Program:

```
T(x):=[x[1]+x[2],x[1]-x[2],x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
```

```
T(x):=[x[1]+x[2],x[1]-x[2],x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
Given Linear Transformation is T(x,y)= [y + x, x - y, y]
Nullity of T= 0
Given linear transformation is Non Singular
```

Output:

Given Linear Transformation is $T(x, y) = [y + x, x - y, y]$
Nullity of $T = 0$
Given linear transformation is Non Singular

Problem 2. Write a program to verify whether the linear transformation $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (0, 0)$ is singular / non-singular

Program:

```
T(x):=[0, 0]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
```

```
T(x):=[0, 0]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
Given Linear Transformation is T(x,y)= [0, 0]
Nullity of T= 2
Given linear transformation is Singular
```

Output:

Given Linear Transformation is $T(x, y) = [0, 0]$
Nullity of $T = 2$
Given linear transformation is Singular

Problem 3. Write a program to verify whether the linear transformation $T: R^3 \rightarrow R^3$ defined by $T(x, y, z) = (x, y, z)$ is singular / non-singular

Program:

```
T(x):=[x[1],x[2],x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
```

```
T(x):=[x[1],x[2],x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
Given Linear Transformation is T(x,y,z)= [x ,y ,z]
Nullity of T= 0
Given linear transformation is Non Singular
```

Output:

Given Linear Transformation is $T(x, y, z) = [x, y, z]$

Nullity of $T = 0$

Given linear transformation is Non Singular

Problem 4. Write a program to verify whether the linear transformation $T: R^4 \rightarrow R^4$ defined by $T(w, x, y, z) = (0, x, y, z)$ is singular / non-singular

Program:

```
T(x):=[0,x[2],x[3],x[4]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
```

```
T(x):=[0,x[2],x[3],x[4]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
Given Linear Transformation is T(w,x,y,z)= [0 ,x ,y ,z]
Nullity of T= 1
Given linear transformation is Singular
```

Output:

Given Linear Transformation is $T(w, x, y, z) = [0, x, y, z]$

Nullity of $T = 1$

Given linear transformation is Singular

Problem 5. Write a program to verify whether the linear transformation $T: \mathbb{R}^4 \rightarrow \mathbb{R}^2$ defined by $T(\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z}) = (\mathbf{w} + \mathbf{x}, \mathbf{y} + \mathbf{z})$ is singular / non-singular

Program:

```
T(x):=[x[1]+x[2],x[3]+x[4]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
```

```
T(x):=[x[1]+x[2],x[3]+x[4]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
Given Linear Transformation is T(w,x,y,z)= [x+w,z+y]
Nullity of T= 2
Given linear transformation is Singular
```

Output:

Given Linear Transformation is $T(\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z}) = [\mathbf{x} + \mathbf{w}, \mathbf{z} + \mathbf{y}]$
Nullity of $T = 2$
Given linear transformation is Singular

Problem 6. Write a program to verify whether the linear transformation $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(\mathbf{x}, \mathbf{y}, \mathbf{z}) = (\mathbf{x} + \mathbf{y}, \mathbf{y} + \mathbf{z}, \mathbf{z} + \mathbf{x})$ is singular / non-singular

Program:

```
T(x):=[x[1]+x[2], x[2]+x[3], x[3]+x[1]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
```

```
T(x):=[x[1]+x[2], x[2]+x[3], x[3]+x[1]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Nullity of T=",nullity(M))$
if nullity(M)=0 then
print("Given linear transformation is Non Singular") else
print("Given linear transformation is Singular")$
Given Linear Transformation is T(x,y,z)= [y+x,z+y,z+x]
Nullity of T= 0
Given linear transformation is Non Singular
```

Output:

Given Linear Transformation is $T(\mathbf{x}, \mathbf{y}, \mathbf{z}) = [\mathbf{y} + \mathbf{x}, \mathbf{z} + \mathbf{y}, \mathbf{z} + \mathbf{x}]$
Nullity of $T = 0$
Given linear transformation is Non Singular

Exercise:

Write a program to verify if the given linear transformation is singular / non-singular

1. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (x, y)$

(Answer: *Non Singular*)

2. $T: \mathbb{R}^4 \rightarrow \mathbb{R}^3$ defined by $T(w, x, y, z) = (2w, x, y + z)$

(Answer: *Singular*)

3. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (x + z, -x + 2y + z, y + z)$

(Answer: *Singular*)

4. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (-2z + y + x, z + 2y + x, -3z + 2y + 2x)$

(Answer: *Non Singular*)

5. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defined by $T(x, y, z) = (x + y, y + z)$

(Answer: *Singular*)

6. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (x + y, x - y)$

(Answer: *Non Singular*)

7. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (y, x)$

(Answer: *Non Singular*)

8. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (x + 2y, 0)$

(Answer: *Singular*)

9. $T: \mathbb{R}^2 \rightarrow \mathbb{R}$ defined by $T(x, y) = x + y$

(Answer: *Singular*)

10. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defined by $T(x, y) = (x, y, x + y)$

(Answer: *Non Singular*)

Program 9

Program to find the minimal polynomial of given linear transformation.

Aim: To find the minimal polynomial of given linear transformation / matrix using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
load("diag")	Loads Package diag.
* (asterisk)	The operator * represents commutative multiplication
[a ₁ , a ₂ , ..., a _m]	List of numbers/objects a ₁ , a ₂ , ..., a _m .
L[i]	Returns i-th element of the list L
'	The single quote operator ' prevents evaluation.
and	The logical conjunction operator
length (expr)	Returns the number of parts in the external (displayed) form of <i>expr</i>
:=	The function definition operator
if cond_1 then expr_1 else expr_0	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0.
print ("text", expr)\$	Displays <i>text</i> within inverted commas and evaluates and displays <i>expr</i>
transpose (M)	Returns the transpose of <i>M</i> .
apply(matrix,L)	Converting nested lists L to matrix
expand (expr)	Expand expression <i>expr</i> .
factor (expr)	Factors the expression <i>expr</i> , containing any number of variables or functions, into factors irreducible over the integers
charpoly (M, x)	Returns the characteristic polynomial for the matrix <i>M</i> with respect to variable <i>x</i> .
minimalPoly (l)	Returns the minimal polynomial of the matrix whose Jordan form is described by the list <i>l</i> .
jordan (mat)	Returns the Jordan form of matrix <i>mat</i> , encoded as a list in a particular format.

- Note:1. Press Shift+Enter for evaluation of commands and display of output.
2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
3. Replace dollar (\$) by semicolon (;) to see output of any input line.
4. Start each session with kill(all)\$ or quit())\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Characteristic Polynomial of a matrix / linear transformation: Let M be an $n \times n$ matrix. $f(\lambda) = \det(M - \lambda I)$ is a polynomial in λ of degree n and is called the **Characteristic Polynomial** of M and $f(\lambda) = 0$ is called the **Characteristic Equation**. By Cayley-Hamilton Theorem, every square matrix satisfies its characteristic equation i.e., $f(M) = 0$. The characteristic polynomial / equation of a linear transformation $T: U \rightarrow U$ is the characteristic polynomial / equation of its associated matrix. The roots of the characteristic equation of a matrix / linear transformation are precisely its eigen values.

Monic Polynomial: A monic polynomial is a non-zero univariate polynomial (that is, a polynomial in a single variable) in which the leading coefficient (the nonzero coefficient of highest degree) is equal to 1. Thus, $f(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_1 x + a_0$ will be a monic polynomial of degree n if $a_n = 1$. For example, $x^2 + 2x + 1$ and $x^3 - 4x^2 + 2x + 1$ are monic polynomials but $2x^2 + 3x + 1$ is not a monic polynomial.

Minimal Polynomial of a matrix / linear transformation: A monic polynomial of the smallest degree which is satisfied by the given matrix or linear transformation is called its minimal polynomial. A polynomial $f(x)$ is the minimal polynomial of the matrix M / Linear transformation T then:

1. $f(x)$ is monic polynomial
2. $f(M) = 0$ / $f(T) = 0$
3. If $g(M) = 0$ / $g(T) = 0$ then $\deg f(x) \leq \deg g(x)$
4. If $\deg h(x) < \deg f(x)$ then $h(M) \neq 0$ / $h(T) \neq 0$

Program:

Program to find the characteristic polynomial and the minimal polynomial of a matrix M

```
load("diag")$
M:matrix([R1],[R2],[R3],...,[Rn])$
C:factor(charpoly(M,x))$
P:minimalPoly(jordan(M))$
print("Given Matrix is M=",M)$
print("Characteristic Polynomial of M=",C,"=",expand(C))$
print("Minimal Polynomial of M=",P,"=",expand(P))$
```

Note: sometimes `C:factor(-charpoly(M,x))$` is used to get positive leading coefficient

Program to the characteristic polynomial and the minimal polynomial of a linear transformation $T: U \rightarrow U$

```
load("diag")$
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
B: Standard ordered basis of domain$
L: makelist(T(B[i]),i,1,length(B))$
M: transpose(apply('matrix,L))$
C: factor(-charpoly(M,x))$
P: minimalPoly(jordan(M))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Characteristic Polynomial of T=",C,"=",expand(C))$
print("Minimal Polynomial of T=",P,"=",expand(P))$
```

Note: 1. In the above program $X:[x,y,z]$ is taken for illustration.

Take $X:[x,y]$ if $U = R^2$, $X:[x,y,z]$ if $U = R^3$, $X:[w,x,y,z]$ if $U = R^4$ and so on.

2. Take standard basis as: $[[1,0],[0,1]]$ for R^2 , $[[1,0,0],[0,1,0],[0,0,1]]$ for R^3

$[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$ for R^4 and so on.

Worked Examples:

Problem 1. Write a program to find the characteristic polynomial and the minimal polynomial of

the matrix $M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$

Program:

```
load("diag")$
M: matrix([0,1,0,1],[1,0,1,0],[0,1,0,1],[1,0,1,0])$
C: factor(charpoly(M,x))$
P: minimalPoly(jordan(M))$
print("Given Matrix is M=",M)$
print("Characteristic Polynomial of M=",C,"=",expand(C))$
print("Minimal Polynomial of M=",P,"=",expand(P))$
```

```
load("diag")$
M: matrix([0,1,0,1],[1,0,1,0],[0,1,0,1],[1,0,1,0])$
C: factor(charpoly(M,x))$
P: minimalPoly(jordan(M))$
print("Given Matrix is M=",M)$
print("Characteristic Polynomial of M=",C,"=",expand(C))$
print("Minimal Polynomial of M=",P,"=",expand(P))$
```

Given Matrix is $M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$

Characteristic Polynomial of $M = (x-2)x^2(x+2) = x^4 - 4x^2$

Minimal Polynomial of $M = (x-2)x(x+2) = x^3 - 4x$

Output:

$$\text{Given Matrix is } M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\text{Characteristic Polynomial of } M = (x - 2)x^2(x + 2) = x^4 - 4x^2$$

$$\text{Minimal Polynomial of } M = (x - 2)x(x + 2) = x^3 - 4x$$

Problem 2. Write a program to find the characteristic polynomial and the minimal polynomial of

the matrix $M = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 2 & -1 \\ -1 & 1 & 3 \end{pmatrix}$

Program:

```
load("diag")$
M:matrix([2,1,1],[-1,2,-1],[-1,1,3])$
C:factor(-charpoly(M,x))$
P:radcan(minimalPoly(jordan(M)))$
print("Given Matrix is M=",M)$
print("Characteristic Polynomial of M=",C)$
print("Minimal Polynomial of M=",P)$
```

```
load("diag")$
M:matrix([2,1,1],[-1,2,-1],[-1,1,3])$
C:factor(-charpoly(M,x))$
P:radcan(minimalPoly(jordan(M)))$
print("Given Matrix is M=",M)$
print("Characteristic Polynomial of M=",C)$
print("Minimal Polynomial of M=",P)$
```

$$\text{Given Matrix is } M = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 2 & -1 \\ -1 & 1 & 3 \end{pmatrix}$$
$$\text{Characteristic Polynomial of } M = x^3 - 7x^2 + 19x - 19$$
$$\text{Minimal Polynomial of } M = x^3 - 7x^2 + 19x - 19$$

Output:

$$\text{Given Matrix is } M = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 2 & -1 \\ -1 & 1 & 3 \end{pmatrix}$$

$$\text{Characteristic Polynomial of } M = x^3 - 7x^2 + 19x - 19$$

$$\text{Minimal Polynomial of } M = x^3 - 7x^2 + 19x - 19$$

Problem 3. Write a program to find the characteristic polynomial and the minimal polynomial of the linear transformation $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x + y, y + z, z + x)$

Program:

```
load("diag")$
T(x):=[x[1]+x[2],x[2]+x[3],x[3]+x[1]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
```

```

C:factor(-charpoly(M,x))$
P:minimalPoly(jordan(M))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Characteristic Polynomial of T=",C,"=",expand(C))$
print("Minimal Polynomial of T=",P,"=",expand(P))$

```

Output:

Given Linear Transformation is $T(x, y, z) = [y + x, z + y, z + x]$

Characteristic Polynomial of $T = (x - 2)(x^2 - x + 1) = x^3 - 3x^2 + 3x - 2$

Minimal Polynomial of $T = (x - 2) \left(x + \frac{\sqrt{3}i - 1}{2} \right) \left(x - \frac{\sqrt{3}i + 1}{2} \right) = x^3 - 3x^2 + 3x - 2$

```

load("diag")$
T(x):=[x[1]+x[2],x[2]+x[3],x[3]+x[1]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
C:factor(-charpoly(M,x))$
P:minimalPoly(jordan(M))$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Characteristic Polynomial of T=",C,"=",expand(C))$
print("Minimal Polynomial of T=",P,"=",expand(P))$

Given Linear Transformation is T(x,y,z)= [y+x ,z+y ,z+x]

Characteristic Polynomial of T= (x-2)(x^2-x+1) = x^3-3x^2+3x-2

Minimal Polynomial of T= (x-2) \left( x + \frac{\sqrt{3}i-1}{2} \right) \left( x - \frac{\sqrt{3}i+1}{2} \right) = x^3-3x^2+3x-2

```

Problem 4. Write a program to find the characteristic polynomial and the minimal polynomial of the linear transformation $T: \mathbf{R}^4 \rightarrow \mathbf{R}^4$ defined by

$$T(w, x, y, z) = (w, w + x, w + x + y, w + x + y + z)$$

Program:

```

load("diag")$
T(x):=[x[1],x[1]+x[2],x[1]+x[2]+x[3],x[1]+x[2]+x[3]+x[4]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
C:factor(charpoly(M,x))$
P:minimalPoly(jordan(M))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Characteristic Polynomial of T=",C,"=",expand(C))$
print("Minimal Polynomial of T=",P,"=",expand(P))$

```

Output:

Given Linear Transformation is $T(w, x, y, z) = [w, x + w, y + x + w, z + y + x + w]$

Characteristic Polynomial of $T = (x - 1)^4 = x^4 - 4x^3 + 6x^2 - 4x + 1$

Minimal Polynomial of $T = (x - 1)^4 = x^4 - 4x^3 + 6x^2 - 4x + 1$

```
load("diag")$
T(x):=[x[1],x[1]+x[2],x[1]+x[2]+x[3],x[1]+x[2]+x[3]+x[4]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
C:factor(charpoly(M,x))$
P:minimalPoly(jordan(M))$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
print("Characteristic Polynomial of T=",C,"=",expand(C))$
print("Minimal Polynomial of T=",P,"=",expand(P))$
Given Linear Transformation is T(w,x,y,z)= [w,x+w,y+x+w,z+y+x+w]
Characteristic Polynomial of T= (x-1)^4 = x^4-4x^3+6x^2-4x+1
Minimal Polynomial of T= (x-1)^4 = x^4-4x^3+6x^2-4x+1
```

Problem 5. Write a program to find the characteristic polynomial and the minimal polynomial of the linear transformation $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (x, -y)$

Program:

```
load("diag")$
T(x):=[x[1],-x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
C:factor(charpoly(M,x))$
P:minimalPoly(jordan(M))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Characteristic Polynomial of T=",C,"=",expand(C))$
print("Minimal Polynomial of T=",P,"=",expand(P))$
```

```
load("diag")$
T(x):=[x[1],-x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
C:factor(charpoly(M,x))$
P:minimalPoly(jordan(M))$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Characteristic Polynomial of T=",C,"=",expand(C))$
print("Minimal Polynomial of T=",P,"=",expand(P))$
Given Linear Transformation is T(x,y)= [x,-y]
Characteristic Polynomial of T= (x-1)(x+1) = x^2-1
Minimal Polynomial of T= (x-1)(x+1) = x^2-1
```

Output:

Given Linear Transformation is $T(x, y) = [x, -y]$

Characteristic Polynomial of $T = (x - 1)(x + 1) = x^2 - 1$

Minimal Polynomial of $T = (x - 1)(x + 1) = x^2 - 1$

Exercise:

I. Write a program to find the characteristic polynomial and the minimal polynomial of the matrices:

1. $M = \begin{pmatrix} 3 & 1 & -1 \\ 2 & 2 & -1 \\ 2 & 2 & 0 \end{pmatrix}$ (Answer: $x^3 - 5x^2 + 8x - 4$ and $x^3 - 5x^2 + 8x - 4$)

2. $M = \begin{pmatrix} 5 & -6 & -6 \\ -1 & 4 & 2 \\ 3 & -6 & -4 \end{pmatrix}$ (Answer: $x^3 - 5x^2 + 8x - 4$ and $x^2 - 3x + 2$)

3. $M = \begin{pmatrix} 3 & -1 & 0 \\ 0 & 2 & 0 \\ 1 & -1 & 2 \end{pmatrix}$ (Answer: $x^3 - 7x^2 + 16x - 12$ and $x^2 - 5x + 6$)

4. $M = \begin{pmatrix} 2 & 5 \\ 6 & 1 \end{pmatrix}$ (Answer: $x^2 - 3x - 28$ and $x^2 - 3x - 28$)

II. Write a program to find the characteristic polynomial and the minimal polynomial of the linear transformations:

1. $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (3x + y, 6x + 2y)$

(Answer: $x^2 - 1$ and $x^2 - 1$)

2. $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x, y, z)$

(Answer: $x^3 - 3x^2 + 3x - 1$ and $x - 1$)

3. $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y) = (0, 0, 0)$

(Answer: x^3 and x)

4. $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (0, x, y)$

(Answer: x^3 and x^3)

5. $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (4x + z, 2x + 3y + 2z, x + 4z)$

(Answer: $x^3 - 11x^2 + 39x - 45$ and $x^2 - 8x + 15$)

Program 10

Program to find the algebraic multiplicity and geometric multiplicity of the Eigenvalues of the given linear transformation.

Aim: To find the algebraic multiplicity and geometric multiplicity of the Eigenvalues of the given linear transformation /matrix using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
<code>*</code> (asterisk)	The operator <code>*</code> represents commutative multiplication
<code>^</code>	Exponentiation operator or power or index
<code>[a₁, a₂, ..., a_m]</code>	List of numbers/objects $a_1, a_2, \dots, a_m$.
<code>L[i]</code>	Returns i-th element of the list <code>L</code>
<code>'</code>	The single quote operator <code>'</code> prevents evaluation.
<code>makelist (expr, i, i_0, i_max)</code>	Returns the list of elements obtained when <code>ev (expr, i=j)</code> is applied to the elements <code>j</code> of the sequence: <code>i_0, i_0 + 1, i_0 + 2, ...</code> , with <code> j </code> less than or equal to <code> i_max </code> .
<code>length (expr)</code>	Returns the number of parts in the external (displayed) form of <code>expr</code>
<code>:=</code>	The function definition operator
<code>if cond_1 then expr_1 else expr_0</code>	evaluates to <code>expr_1</code> if <code>cond_1</code> evaluates to true, otherwise the expression evaluates to <code>expr_0</code> .
<code>print ("text", expr)\$</code>	Displays <code>text</code> within inverted commas and evaluates and displays <code>expr</code>
<code>transpose (M)</code>	Returns the transpose of <code>M</code> .
<code>apply('matrix,L)</code>	Converting nested lists <code>L</code> to matrix
<code>reverse (list)</code>	Reverses the order of the members of the <code>list</code> (not the members themselves)
<code>table_form()</code>	Displays a 2D list in a form that is more readable than the output from <i>Maxima</i> 's default output routine. The input is a list of one or more lists.
<code>push (item, list)</code>	<code>push</code> prepends the item <code>item</code> to the list <code>list</code> and returns a copy of the new list.
<code>eigenvalues (M) or eivals (M)</code>	Returns a list of two lists containing the eigenvalues of the matrix <code>M</code> . The first sublist of the return value is the list of eigenvalues of the matrix, and the second sublist is the list of the multiplicities of the eigenvalues in the corresponding order.

eigenvectors (M) or eivects (M)	Computes eigenvectors of the matrix M . The return value is a list of two elements. The first is a list of the eigenvalues of M and a list of the multiplicities of the eigenvalues. The second is a list of lists of eigenvectors. There is one list of eigenvectors for each eigenvalue. There may be one or more eigenvectors in each list.
---	--

Note:1. Press Shift+Enter for evaluation of commands and display of output.

2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.

3. Replace dollar (\$) by semicolon (;) to see output of any input line.

4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Eigen value and Eigen vector of a Linear transformation: Let V be a vector space over a field F and $T: V \rightarrow V$ be a linear transformation / linear operator. A scalar $\lambda \in F$ is called an eigen value / characteristic value of T if $T(v) = \lambda v$ for some non-zero vector $v \in V$. Then v is called an eigen vector / characteristic vector corresponding to the eigen value λ .

Similarly, a non-zero vector $v \in V$ is called an eigen vector / characteristic vector of T if $T(v) = \lambda v$ for some scalar $\lambda \in F$. Then $\lambda \in F$ is called an eigen value / characteristic value corresponding to the eigen vector v .

Algebraic Multiplicity of an Eigen Value: An eigen value of a linear transformation may be repeated. The number of times an eigen value is repeated is called its **algebraic multiplicity**. For a non-repeated eigen value, algebraic multiplicity is 1. For twice repeated eigen value, algebraic multiplicity is 2 and so on. Using characteristic polynomial, algebraic multiplicity of an eigen value λ is the largest integer k for which $(x - \lambda)^k$ is a factor of the characteristic polynomial $f(x)$.

Geometric Multiplicity of an Eigen Value: The number of linearly independent eigen vectors corresponding to an eigen value is called its **geometric multiplicity**. In fact, geometric multiplicity is the dimension of eigen space of an eigen value. For a non-repeated eigen value, geometric multiplicity is 1. For twice repeated eigen value, geometric multiplicity may be 1 or 2 depending on the dimension of its eigen space.

Relation between algebraic and geometric multiplicities of an eigen value: For any eigen value

$$\text{geometric multiplicity} \leq \text{algebraic multiplicity}$$

Program:

Program to find the algebraic multiplicity and geometric multiplicity of eigen values of a linear transformation $T: R^n \rightarrow R^n$

```
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
B:Standard basis of given vector space$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
table_form(reverse(N))$
```

Note: 1. Here X:[x,y,z] is taken for illustration purpose.

Take X:[x,y] if domain is R^2 , X:[x,y,z] if domain is R^3 ,

X:[w,x,y,z] if domain is R^4 and so on.

2. Take standard basis as: B:[[1,0],[0,1]] for R^2 , B:[[1,0,0],[0,1,0],[0,0,1]] for R^3 ,

B: [[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]] for R^4 and so on.

Program to find the algebraic multiplicity and geometric multiplicity of eigen values of a matrix

```
M:matrix([R1],[R2],...,[Rn])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear matrix is M=",M)$
table_form(reverse(N))$
```

Worked Examples:

Problem 1. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values of the linear transformation $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (x + 4y, 2x + 3y)$

Program:

```
T(x):=[x[1]+4*x[2],2*x[1]+3*x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y)=",T(X))$
table_form(reverse(N))$
```

Output:

Given Linear Transformation is $T(x, y) = [4y + x, 3y + 2x]$

Eigen Value	Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 5$	1	$\{[1, 1]\}$
$\lambda_2 = -1$	1	$\left\{ \left[1, -\frac{1}{2} \right] \right\}$

```
T(x):=[x[1]+4*x[2],2*x[1]+3*x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y)=",T(X))$
table_form(reverse(N))$
```

Given Linear Transformation is $T(x, y) = [4y + x, 3y + 2x]$

Eigen Value	algebraic Multiplicity	Geometric Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 5$	1	1	$\{[1, 1]\}$
$\lambda_2 = -1$	1	1	$\left\{ \left[1, -\frac{1}{2} \right] \right\}$

Problem 2. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values of the linear transformation $T: \mathbf{R}^4 \rightarrow \mathbf{R}^4$ defined by $T(w, x, y, z) = (0, w, x, y)$

Program:

```
T(x):=[0,x[1],x[2],x[3]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
table_form(reverse(N))$
```

Output:

Given Linear Transformation is $T(w, x, y, z) = [0, w, x, y]$

<i>Eigen Value</i>	<i>algebraic Multiplicity</i>	<i>Geometric Multiplicity</i>	<i>Linearly Independent Eigen vectors</i>
$\lambda_1 = 0$	4	1	$\{[0,0,0,1]\}$

```
T(x):=[0,x[1],x[2],x[3]]$
X:[w,x,y,z]$
B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(w,x,y,z)=",T(X))$
table_form(reverse(N))$

Given Linear Transformation is T(w,x,y,z)= [0, w, x, y]

Eigen Value algebraic Multiplicity Geometric Multiplicity Linearly Independent Eigen vectors
λ1 =0 4 1 { [0, 0, 0, 1] }
```

Problem 3. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values of the linear transformation $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x, 2x + y, 3x + 2y)$

Program:

```
T(x):=[x[1], 2*x[1]+x[2],3*x[1]+2*x[2]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]
```

```

N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]
for i thru length(vals[1]) do N:push(N(i),N)
print("Given Linear Transformation is T(x,y,z)=",T(X))
table_form(reverse(N))

```

Output:

Given Linear Transformation is $T(x, y, z) = [x, y + 2x, 2y + 3x]$

<i>Eigen Value</i>	<i>algebraic Multiplicity</i>	<i>Geometric Multiplicity</i>	<i>Linearly Independent Eigen vectors</i>
$\lambda_1 = 0$	1	1	{[0,0,1]}
$\lambda_2 = 1$	2	1	{[0,1,2]}

```

T(x):=[x[1], 2·x[1]+x[2],3·x[1]+2·x[2]]
X:[x,y,z]
B:[[1,0,0],[0,1,0],[0,0,1]]
L:makelist(T(B[i]),i,1,length(B))
M:transpose(apply('matrix,L))
[vals, vecs]:eigenvalues(M)
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]
for i thru length(vals[1]) do N:push(N(i),N)
print("Given Linear Transformation is T(x,y,z)=",T(X))
table_form(reverse(N))

Given Linear Transformation is T(x,y,z)= [x ,y +2 x ,2 y +3 x ]

Eigen Value  algebraic Multiplicity  Geometric Multiplicity  Linearly Independent Eigen vectors
λ1 =0        1                        1                        {[0, 0, 1]}
λ2 =1        2                        1                        {[0, 1, 2]}

```

Problem 4. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values of the linear transformation $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by

$$T(x, y, z) = (5x - 6y - 6z, -x + 4y + 2z, 3x - 6y - 4z)$$

Program:

```

T(x):=[5*x[1]-6*x[2]-6*x[3],-x[1]+4*x[2]+2*x[3],3*x[1]-6*x[2]-4*x[3]]
X:[x,y,z]
B:[[1,0,0],[0,1,0],[0,0,1]]
L:makelist(T(B[i]),i,1,length(B))
M:transpose(apply('matrix,L))
[vals, vecs]:eigenvalues(M)
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]
for i thru length(vals[1]) do N:push(N(i),N)
print("Given Linear Transformation is T(x,y,z)=",T(X))
table_form(reverse(N))

```

Output:

Given Linear Transformation is $T(x, y, z) = [-6z - 6y + 5x, 2z + 4y - x, -4z - 6y + 3x]$

Eigen Value algebraic Multiplicity Geometric Multiplicity Linearly Independent Eigen vectors

$\lambda_1 = 1$	1	1	$\left\{ \left[1, -\frac{1}{3}, 1 \right] \right\}$
$\lambda_2 = 2$	2	2	$\left\{ [0, 1, -1], \left[1, 0, \frac{1}{2} \right] \right\}$

```
T(x):=[5*x[1]-6*x[2]-6*x[3],-x[1]+4*x[2]+2*x[3],3*x[1]-6*x[2]-4*x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:=makelist(T(B[i]),i,1,length(B))$
M:=transpose(apply('matrix,L))$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
table_form(reverse(N))$

Given Linear Transformation is T(x,y,z)= [-6 z-6 y+5 x,2 z+4 y-x,-4 z-6 y+3 x]

Eigen Value   algebraic Multiplicity   Geometric Multiplicity   Linearly Independent Eigen vectors

λ1=1                      1                      1                      { [ 1, - 1/3, 1 ] }
```

Problem 5. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values

of the matrix $M = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & -6 \\ 2 & -2 & 3 \end{pmatrix}$

Program:

```
M:=matrix([1,2,0],[2,1,-6],[2,-2,3])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear matrix is M=",M)$
table_form(reverse(N))$
```

Output:

Given Linear matrix is $M = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & -6 \\ 2 & -2 & 3 \end{pmatrix}$

Eigen Value algebraic Multiplicity Geometric Multiplicity Linearly Independent Eigen vectors

$\lambda_1 = -3$	1	1	$\{ [1, -2, -1] \}$
$\lambda_2 = 3$	1	1	$\{ [1, 1, 0] \}$
$\lambda_3 = 5$	1	1	$\{ [1, 2, -1] \}$


```

M:matrix([1,2,0],[2,1,-6],[2,-2,3])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear matrix is M=",M)$
table_form(reverse(N))$

```

Given Linear matrix is $M = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & -6 \\ 2 & -2 & 3 \end{pmatrix}$

Eigen Value	algebraic Multiplicity	Geometric Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = -3$	1	1	$\{[1, -2, -1]\}$
$\lambda_2 = 3$	1	1	$\{[1, 1, 0]\}$
$\lambda_3 = 5$	1	1	$\{[1, 2, -1]\}$

Problem 6. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values

of the matrix $M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$

Program:

```

M:matrix([1,1],[0,1])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear matrix is M=",M)$
table_form(reverse(N))$

```

Output:

Given Linear matrix is $M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$

Eigen Value	algebraic Multiplicity	Geometric Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 1$	2	1	$\{[1, 0]\}$

```

M:matrix([1,1],[0,1])$
[vals, vecs]:eigenvectors(M)$
N:[["Eigen Value","algebraic Multiplicity","Geometric Multiplicity","Linearly Independent Eigen vectors"]]$
N(i):=[λ[i]=vals[1][i],vals[2][i],length(vecs[i]),setify(vecs[i])]$
for i thru length(vals[1]) do N:push(N(i),N)$
print("Given Linear matrix is M=",M)$
table_form(reverse(N))$

```

Given Linear matrix is $M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$

Eigen Value	algebraic Multiplicity	Geometric Multiplicity	Linearly Independent Eigen vectors
$\lambda_1 = 1$	2	1	$\{[1, 0]\}$

Exercise:

I. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values of the given linear transformations:

1. $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ defined by $T(x, y) = (y, 0)$ (Answer: $\lambda = 0$, AM=2, GM=1)

2. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (0, 0, 0)$ (Answer: $\lambda = 0$, AM=3, GM=3)

3. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (x, y, z)$ (Answer: $\lambda = 1$, AM=3, GM=3)

4. $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ defined by $T(x, y, z) = (x, x - y, x - z)$

(Answer: $\lambda = 1$, AM=1, GM=1, $\lambda = -1$, AM=2, GM=2)

5. $T: \mathbb{R}^4 \rightarrow \mathbb{R}^4$ defined by $T(w, x, y, z) = (w + x, x + y, y + z, z + w)$

(Answer: $\lambda = 0$, AM=1, GM=1, $\lambda = 2$, AM=1, GM=1, $\lambda = 1 + i$, AM=1, GM=1, $\lambda = 1 - i$, AM=1, GM=1)

II. Write a program to find the algebraic multiplicity and geometric multiplicity of eigen values of the given matrices:

1. $M = \begin{pmatrix} 2 & 5 \\ 0 & 2 \end{pmatrix}$ (Answer: $\lambda = 2$, AM=2, GM=1)

2. $M = \begin{pmatrix} 0 & 0 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & -2 \end{pmatrix}$ (Answer: $\lambda = 1$, AM=1, GM=1, $\lambda = -1$, AM=1, GM=1, $\lambda = -2$, AM=1, GM=1)

3. $M = \begin{pmatrix} 2 & 4 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 4 \end{pmatrix}$ (Answer: $\lambda = 2$, AM=2, GM=1, $\lambda = 4$, AM=1, GM=1)

4. $M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ -2 & -2 & 2 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix}$ (Answer: $\lambda = 1$, AM=2, GM=1, $\lambda = 0$, AM=2, GM=1)

Program 11

Program on diagonalization.

Aim: To test whether a square matrix / a linear operator is diagonalizable or not and finding its diagonal form if exists using Mathematics Softwares (FOSS).

Software: Maxima

Keys:

Key	Function
load ("eigen")	Loads the package eigen which contains several functions devoted to the symbolic computation of eigenvalues and eigenvectors.
*	The operator * represents commutative multiplication
$\wedge$	Exponentiation operator or power or index
$[a_1, a_2, \dots, a_m]$	List of numbers/objects $a_1, a_2, \dots, a_m$.
$L[i]$	Returns i-th element of the list L
'	The single quote operator ' prevents evaluation.
$\text{makelist} (expr, i, i_0, i_max)$	Returns the list of elements obtained when $\text{ev} (expr, i=j)$ is applied to the elements j of the sequence: $i_0, i_0 + 1, i_0 + 2, \dots$, with $ j $ less than or equal to $ i_max $.
$\text{length} (expr)$	Returns the number of parts in the external (displayed) form of $expr$
$:=$	The function definition operator
$\text{if cond_1 then expr_1 else expr_0}$	evaluates to expr_1 if cond_1 evaluates to true, otherwise the expression evaluates to expr_0 .
$\text{print} ("text", expr)\$$	Displays text within inverted commas and evaluates and displays $expr$
$\text{transpose} (M)$	Returns the transpose of M .
$\text{apply}('matrix, L)$	Converting nested lists L to matrix
$\text{similaritytransform} (M)$	$\text{similaritytransform}$ computes a similarity transform of the matrix M.
leftmatrix	If $P^{-1}MP=D$ is similarity transform of M then leftmatrix refers to P^{-1}
rightmatrix	If $P^{-1}MP=D$ is similarity transform of M then rightmatrix refers to P
$\neq$	Not equal to
$[]$	Empty list
$\text{radcan} (expr)$	Simplifies $expr$, which can contain logs, exponentials, and radicals

- Note:1. Press Shift+Enter for evaluation of commands and display of output.
2. Replace semicolon (;) by dollar (\$) to suppress output of any input line.
 3. Replace dollar (\$) by semicolon (;) to see output of any input line.
 4. Start each session with kill(all)\$ or quit()\$ to remove previously assigned values of all symbols

Definitions and Formulae:

Diagonalization of a square matrix: Let M be a square matrix of order n . M is said to be diagonalizable if M is similar to a diagonal matrix D , that is, $P^{-1}MP = D$ is a diagonal matrix for some non-singular matrix P . Here D is called the diagonal form of M .

Not all square matrices are diagonalizable. A square matrix is diagonalizable if for each eigen value, algebraic multiplicity = geometric multiplicity.

Diagonalization of a linear operator/transformation: Let $T: V \rightarrow V$ be a linear operator and V is a finite-dimensional vector space. T is said to be diagonalizable if there exists a basis $B = \{v_1, v_2, \dots, v_n\}$ of V such that the matrix of T with respect to the basis B is diagonal.

Not all linear operators are diagonalizable. A linear operator is diagonalizable if and only if there exists a basis of V containing eigen vectors of T .

Program:

Program to verify diagonalizability of a matrix M .

Also finding P and D such that $P^{-1}MP = D$ if M is diagonalizable.

```
load("eigen")$
M:matrix([R1],[R2],...,[Rn])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

Program to verify diagonalizability of a linear operator $T: V \rightarrow V$. If T is diagonalizable, finding diagonal form of the matrix of T .

```
T(x):= define T as an ordered list as given in problem in terms of x[i]$
X:[x,y,z]$
B:standard basis of domain$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Matrix of T is M=",M)$
if P≠[] then print("Given T is Diagonalizable") and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given T is not Diagonalizable")$
```

Note: 1. Here $X:[x,y,z]$ is taken for illustration purpose.

Take $X:[x,y]$ if domain is R^2 , $X:[x,y,z]$ if domain is R^3 ,
 $X:[w,x,y,z]$ if domain is R^4 and so on.

- 2.** Take standard basis as: $B:[[1,0],[0,1]]$ for R^2 , $B:[[1,0,0],[0,1,0],[0,0,1]]$ for R^3 ,
 $B:[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]$ for R^4 and so on.

Worked Examples:

Problem 1. Write a program to verify diagonalizability of matrix $M = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

Also find P and D such that $P^{-1}MP = D$ if M is diagonalizable

Program:

```
load("eigen")$
M:matrix([0,-1],[1,0])$
similaritytransform (M)$
P:rightmatrix$
```

```
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

Output:

$$\text{Given matrix is } M = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Given matrix is Diagonalizable

$$\text{Matrix } P = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{\%i}{\sqrt{2}} & -\frac{\%i}{\sqrt{2}} \end{pmatrix}$$

$$\text{Diagonal form of } M \text{ is } D = \begin{pmatrix} -\%i & 0 \\ 0 & \%i \end{pmatrix}$$

Problem 2. Write a program to verify diagonalizability of matrix $M = \begin{pmatrix} -1 & 2 & 4 \\ -2 & 4 & 2 \\ -4 & 2 & 7 \end{pmatrix}$

Also find P and D such that $P^{-1}MP = D$ if M is diagonalizable

Program:

```
load("eigen")$
M:matrix([-1,2,4],[-2,4,2],[-4,2,7])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

Output:

$$\text{Given matrix is } M = \begin{pmatrix} -1 & 2 & 4 \\ -2 & 4 & 2 \\ -4 & 2 & 7 \end{pmatrix}$$

Given matrix is Diagonalizable

```
load("eigen")$
M:matrix([0,-1],[1,0])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

Given matrix is $M = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

Given matrix is Diagonalizable

Matrix $P = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{\%i}{\sqrt{2}} & -\frac{\%i}{\sqrt{2}} \end{pmatrix}$

Diagonal form of M is $D = \begin{pmatrix} -\%i & 0 \\ 0 & \%i \end{pmatrix}$

```
load("eigen")$
M:matrix([-1,2,4],[-2,4,2],[-4,2,7])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

Given matrix is $M = \begin{pmatrix} -1 & 2 & 4 \\ -2 & 4 & 2 \\ -4 & 2 & 7 \end{pmatrix}$

Given matrix is Diagonalizable

Matrix $P = \begin{pmatrix} \frac{2}{3} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{3} & 0 & \frac{2}{\sqrt{5}} \\ \frac{2}{3} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{5}} \end{pmatrix}$

Diagonal form of M is $D = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}$

$$\text{Matrix } P = \begin{pmatrix} \frac{2}{3} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{3} & 0 & \frac{2}{\sqrt{5}} \\ \frac{2}{3} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{5}} \end{pmatrix}$$

$$\text{Diagonal form of } M \text{ is } D = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

Problem 3. Write a program to verify diagonalizability of matrix $M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$

Also find P and D such that $P^{-1}MP = D$ if M is diagonalizable

Program:

```
load("eigen")$
M:matrix([0,1,0,1],[1,0,1,0],[0,1,0,1],[1,0,1,0])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

Output:

$$\text{Given matrix is } M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Given matrix is Diagonalizable

$$\text{Matrix } P = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

```
load("eigen")$
M:matrix([0,1,0,1],[1,0,1,0],[0,1,0,1],[1,0,1,0])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

$$\text{Given matrix is } M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Given matrix is Diagonalizable

$$\text{Matrix } P = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$$\text{Diagonal form of } M \text{ is } D = \begin{pmatrix} -2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\text{Diagonal form of } M \text{ is } D = \begin{pmatrix} -2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Problem 4. Write a program to verify diagonalizability of matrix $M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$

Also find P and D such that $P^{-1}MP = D$ if M is diagonalizable

Program:

```
load("eigen")$
M:matrix([0,-1],[1,0])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

```
load("eigen")$
M:matrix([1,1],[0,1])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$

Given matrix is M=  $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ 
Given matrix is not Diagonalizable
```

Output:

Given matrix is $M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$
 Given matrix is not Diagonalizable

Problem 5. Write a program to verify diagonalizability of matrix $M = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{pmatrix}$

Also find P and D such that $P^{-1}MP = D$ if M is diagonalizable

Program:

```
load("eigen")$
M:matrix([2,1,0],[0,2,1],[0,0,2])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$
```

```
load("eigen")$
M:matrix([2,1,0],[0,2,1],[0,0,2])$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given matrix is M=",M)$
if P≠[] then print("Given matrix is Diagonalizable")
and print("Matrix P=",P) and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given matrix is not Diagonalizable")$

Given matrix is M=  $\begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{pmatrix}$ 
Given matrix is not Diagonalizable
```


Output:

$$\text{Given matrix is } M = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{pmatrix}$$

Given matrix is not Diagonalizable

Problem 6. Write a program to verify diagonalizability of linear operator $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by

$T(x, y, z) = (4x + z, 2x + 3y + 2z, x + 4z)$. Also, find diagonal form of the matrix of T if T is diagonalizable.

Program:

```
T(x):=[4*x[1]+x[3],2*x[1]+3*x[2]+2*x[3],x[1]+4*x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Matrix of T is M=",M)$
if P#[] then print("Given T is Diagonalizable") and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given T is not Diagonalizable")$
```

```
T(x):=[4*x[1]+x[3],2*x[1]+3*x[2]+2*x[3],x[1]+4*x[3]]$
X:[x,y,z]$
B:[[1,0,0],[0,1,0],[0,0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given Linear Transformation is T(x,y,z)=",T(X))$
print("Matrix of T is M=",M)$
if P#[] then print("Given T is Diagonalizable") and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given T is not Diagonalizable")$

Given Linear Transformation is T(x,y,z)= [z+4 x, 2 z+3 y+2 x, 4 z+x]

Matrix of T is M=

$$\begin{pmatrix} 4 & 0 & 1 \\ 2 & 3 & 2 \\ 1 & 0 & 4 \end{pmatrix}$$


Given T is Diagonalizable

Diagonal form of M is D=

$$\begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

```

Output:

Given Linear Transformation is $T(x, y, z) = [z + 4x, 2z + 3y + 2x, 4z + x]$

$$\text{Matrix of } T \text{ is } M = \begin{pmatrix} 4 & 0 & 1 \\ 2 & 3 & 2 \\ 1 & 0 & 4 \end{pmatrix}$$

Given T is Diagonalizable

$$\text{Diagonal form of } M \text{ is } D = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

Problem 7. Write a program to verify diagonalizability of linear operator $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by

$T(x, y) = (2x + y, 2y)$. Also, find diagonal form of the matrix of T if T is diagonalizable.

Program:

```
T(x):=[2*x[1]+x[2],2*x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Matrix of T is M=",M)$
if P≠[] then print("Given T is Diagonalizable") and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given T is not Diagonalizable")$
```

```
T(x):=[2·x[1]+x[2],2·x[2]]$
X:[x,y]$
B:[[1,0],[0,1]]$
L:makelist(T(B[i]),i,1,length(B))$
M:transpose(apply('matrix,L))$
similaritytransform (M)$
P:rightmatrix$
P1:leftmatrix$
print("Given Linear Transformation is T(x,y)=",T(X))$
print("Matrix of T is M=",M)$
if P≠[] then print("Given T is Diagonalizable") and
print("Diagonal form of M is D=",radcan(P1.M.P)) else
print("Given T is not Diagonalizable")$

Given Linear Transformation is T(x,y)= [y + 2 x, 2 y]

Matrix of T is M=  $\begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}$ 

Given T is not Diagonalizable
```

Output:

Given Linear Transformation is $T(x, y) = [y + 2x, 2y]$

Matrix of T is $M = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}$

Given T is not Diagonalizable

Exercise:

I. Write a program to verify diagonalizability of a given matrix M . Also finding P and D such that $P^{-1}MP = D$ if M is diagonalizable.

1. $M = \begin{pmatrix} 5 & 0 \\ 3 & 5 \end{pmatrix}$

(Answer: *Not Diagonalizable*)

2. $M = \begin{pmatrix} 5 & 0 \\ 3 & 6 \end{pmatrix}$

(Answer: *Diagonalizable*)

3. $M = \begin{pmatrix} 6 & 3 & -8 \\ 0 & -2 & 0 \\ 1 & 0 & -3 \end{pmatrix}$

(Answer: *Not diagonalizable*)

4. $M = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}$

(Answer: *Not Diagonalizable*)

5. $M = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 0 \end{pmatrix}$

(Answer: *Not diagonalizable*)

6. $M = \begin{pmatrix} 2 & 0 & 0 \\ 2 & 6 & 0 \\ 3 & 2 & 1 \end{pmatrix}$

(Answer: *Diagonalizable*)

7. $M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

(Answer: *Not Diagonalizable*)

II. Write a program to verify diagonalizability of given linear operator $T: V \rightarrow V$.

Also, the find diagonal form of the matrix of T if T is diagonalizable.

1. $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (0, 0)$

(Answer: *Diagonalizable*)

2. $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x, y, z)$

(Answer: *Diagonalizable*)

3. $T: \mathbf{R}^4 \rightarrow \mathbf{R}^4$ defined by $T(w, x, y, z) = (0, w, x, y)$

(Answer: *Not diagonalizable*)

4. $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (x + y, 2y)$

(Answer: *Diagonalizable*)

5. $T: \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by $T(x, y) = (x + 2y, y)$

(Answer: *Not diagonalizable*)

6. $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (x + 3y, 2x - y, y - z)$

(Answer: *Diagonalizable*)

7. $T: \mathbf{R}^3 \rightarrow \mathbf{R}^3$ defined by $T(x, y, z) = (z, y, x)$

(Answer: *Diagonalizable*)



BLDEA'S S.B. ARTS AND K.C.P. SCIENCE

COLLEGE, VIJAYAPUR

(Affiliated to Rani Channamma University, Belagavi)

Practical's on DSC 2
NUMERICAL ANALYSIS

(AS PER NEP-2020)

MANUAL

B. Sc. VI Semester

Name:.....

Class:.....

UUCMS No:.....

DEPARTMENT OF MATHEMATICS

CONTENTS

Prog. No.	TITLE	No.
1	To find a real root of the given equation using Bisection Method	02
2	To find a real root of the given equation using Regula Falsi Method.	04
3	To find a real root of the given equation using Newton Raphson Method.	06
4	Solving system of equations by Gauss elimination method.	08
5	Solving system of equations by Jacobi iteration method.	10
6	Solving system of equations by Gauss - Seidel iterative method.	13
7	To find the value of $f(x)$ at any point using Newton-Gregory forward interpolation formula.	16
8	Lagrange's Interpolation.	18
9	To evaluate definite integrals using Simpson's one-third rule.	20
10	To evaluate definite integrals using Simpson's three-eight rule.	22

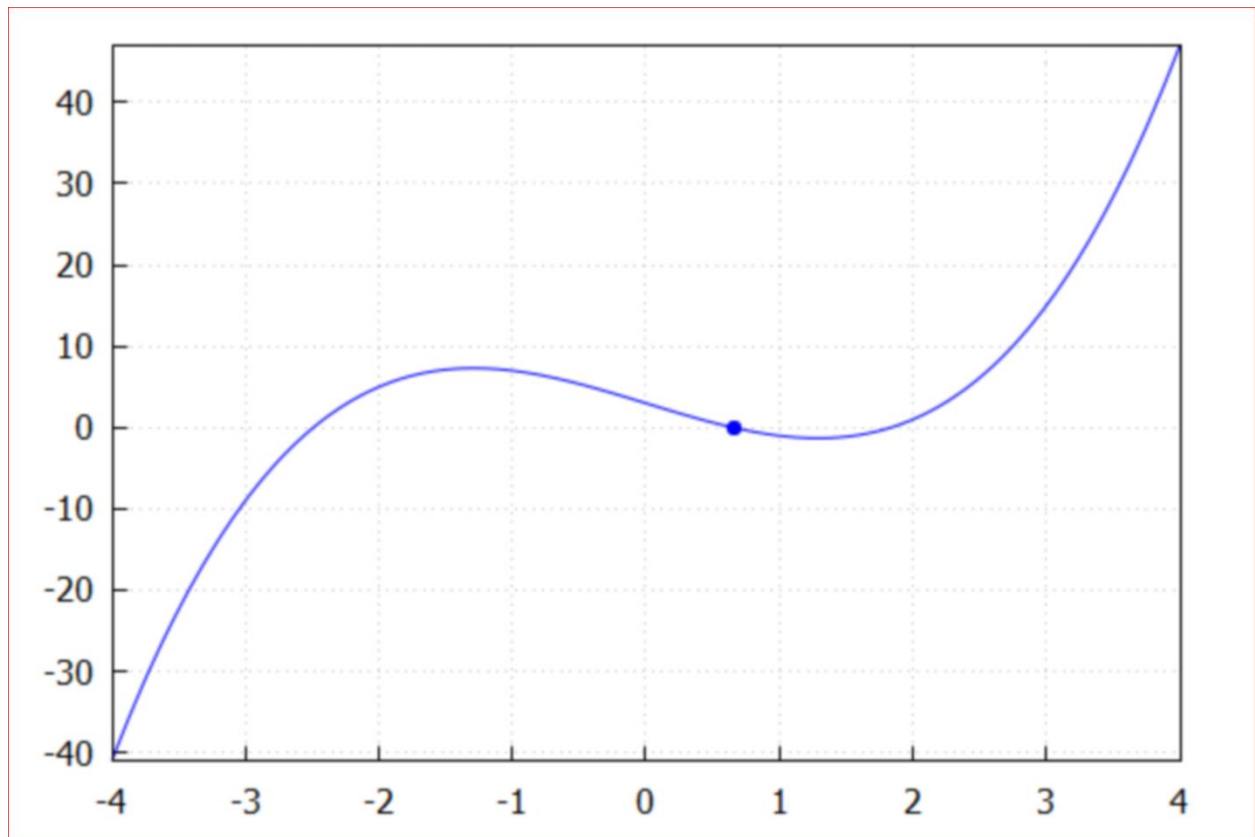
PROGRAM-1: To find a real root of the given equation using Bisection Method.

The method consists of locating the root of the equation $f(x)=0$ between a and b ($a < b$). If $f(x)$ is continuous in the interval $[a, b]$ and $f(a)$ and $f(b)$ are of opposite signs then there is a root between a and b . For definiteness, $f(a)$ be negative and $f(b)$ be positive. Then the first approximation to the root is $x_1 = \frac{a+b}{2}$. If $f(x_1) = 0$ then x_1 is a root of $f(x)=0$. Otherwise, the root lies between a and x_1 or x_1 and b accordingly as $f(x_1)$ is positive or negative. Then we bisect the interval as before and continue the process until the root is found to the desired accuracy.

Example: Maxima program to find a real root of $x^3 - 9x + 1 = 0$ using Bisection method.

```
(%i6) kill(all)$
      fpprintprec:4$
      f(x):=x^3-9*x+1;
      a:read("Enter the lower limit of the interval a=")$
      b:read("Enter the upper limit of the interval b=")$
      if f(a)*f(b)>0 then
      print("Change the limits")
      else (
      for i:1 thru 20 do (
        c:(a+b)/2,
        if f(c)*f(b)>0 then
        b:c
        else
        a:c),
      print("Root is", float(c))$
      wxdraw2d(grid = true,
        explicit(f(x),x,-4,4),
        color=red,
        point_type=filled_circle,
        points([c],[0])
      )$
```

```
(%o2) f(x) := x^3 - 9 x + 1
      Enter the lower limit of the interval a= 0;
      Enter the upper limit of the interval b= 1;
      Root is 0.1113
```



Exercise: Write a maxima program to find a real root of the following equations using Bisection method.

1.1 $x^3 - 5x + 3 = 0$ (Ans: 0. 657)

1.2 $x^3 - x - 1 = 0$ (Ans: 1. 325)

1.3 $x^3 - 16x^2 + 3 = 0$ (Ans: 0.439)

1.4 $xe^x = 1$ (Ans: 0. 567)

1.5 $\cos x - 3x + 1 = 0$ (Ans: 0. 607)

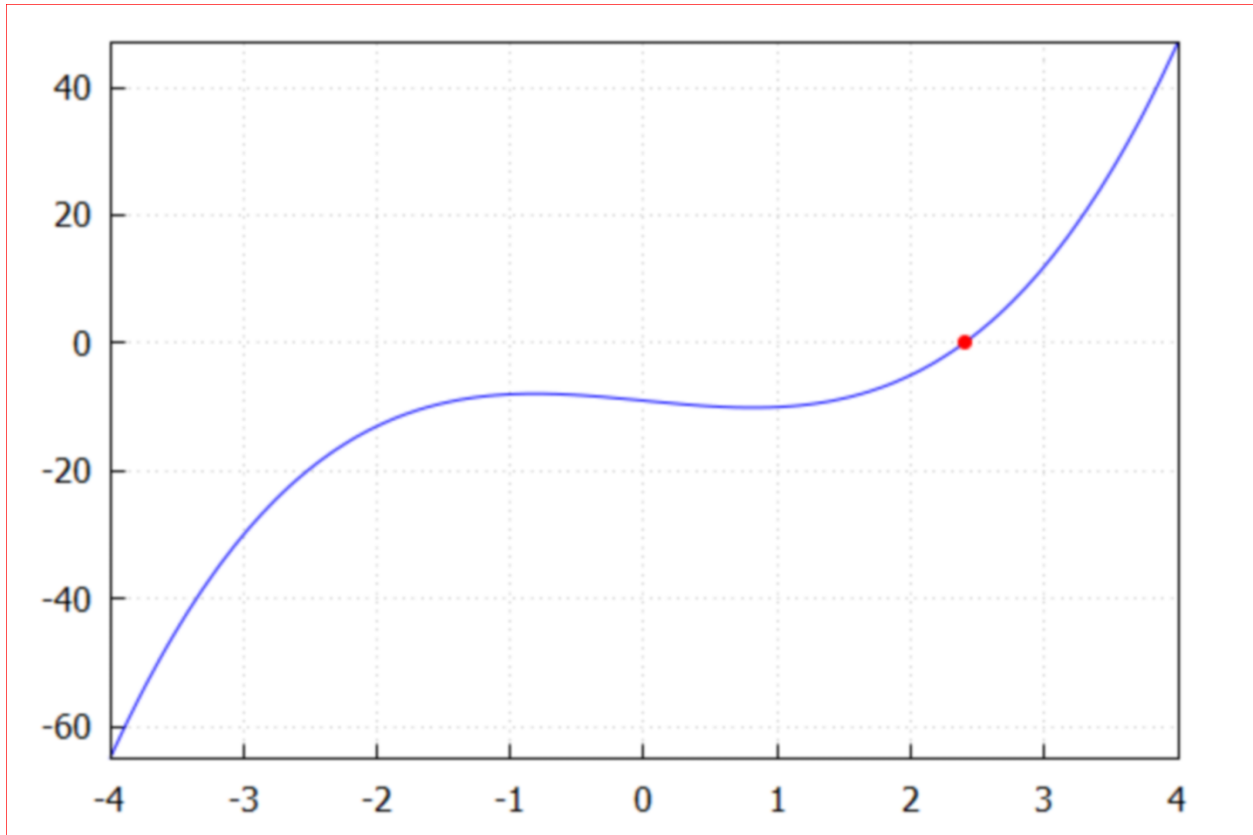
Program-2: To find a real root of the given equation using Regula Falsi Method.

The Regula- Falsi method is based on replacing the part of the curve between the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ by the chord joining these two points and then taking the point of intersection of the chord with x -axis as an approximation to the root. We obtain $x = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)}$, which gives the first approximation. Using this equation, we get a sequence of approximations till we get the root to the desired accuracy.

Example: Maxima program to find a real root of $x^3 - 2x - 9 = 0$ using Regula Falsi method.

```
(%i6) kill(all)$
      fpprintprec:4$
      f(x):=x^3-2*x-9;
      a:read("Enter the lower limit of the interval a=")$
      b:read("Enter the upper limit of the interval b=")$
      if f(a)*f(b)>0 then
      print("Change the limits")
      else (
      for i:1 thru 10 do (
        c:(a*f(b)-b*f(a))/(f(b)-f(a)),
        if f(c)*f(b)>0 then
          b:c
        else
          a:c),
      print("Root is", float(c))$
      wxdraw2d(grid = true,
        explicit(f(x),x,-4,4),
        color=red,
        point_type=filled_circle,
        points([c],[0])
      )$

(%o2) f(x) := x^3 - 2 x - 9
      Enter the lower limit of the interval a= 2;
      Enter the upper limit of the interval b= 3;
      Root is 2.398
```

Exercise: Write a maxima program to find a real root of the following equations using Regula Falsi Method.

2.1 $x^3 - x - 4 = 0$ (Ans: 1. 796)

2.2 $x^3 - 2x - 1 = 0$ (Ans: 1. 618)

2.3 $x^3 - 10x - 5 = 0$ (Ans: 3. 388)

2.4 $x^3 + 4x^2 - 10 = 0$ (Ans: 1. 365)

2.5 $\cos x - 3x + 1 = 0$ (Ans: 0. 607)

Program-03: To find a real root of the given equation using Newton Raphson Method.

Assuming that x_0 is an approximate value of a real root of the equation $f(x) = 0$, let x_1 be the exact root and $x_1 = x_0 + h$, where h is a small correction. Using Taylor's expansion and neglecting higher powers of $h(h^2, h^3, \dots)$, we get

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

In general, $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, $n = 0, 1, 2, \dots$

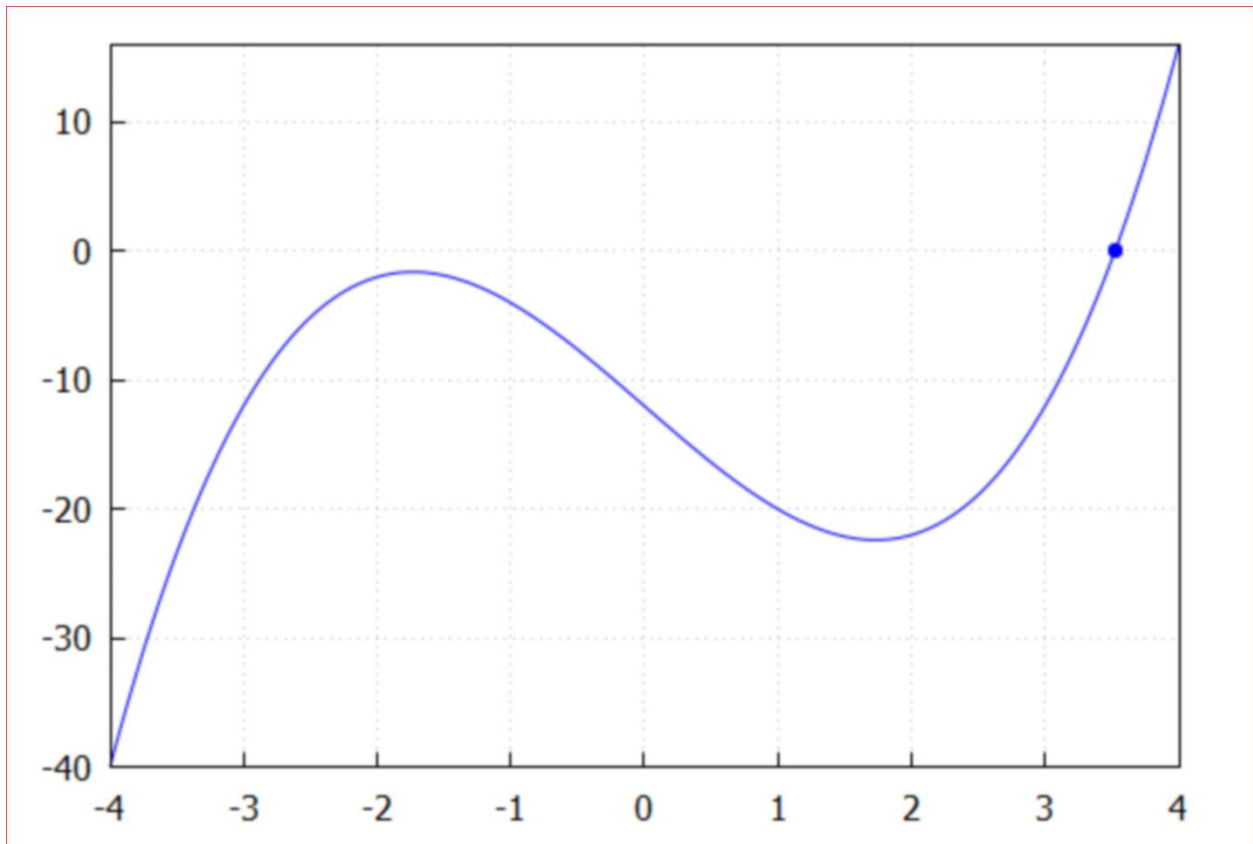
This is Newton-Raphson iterative formula.



Example: Maxima program to find a real root of $x^3 - 9x - 12 = 0$ using Newton Raphson method.

```
(%i7) kill(all)$
      load(newton1)$
      fpprintprec:4$
      f(x):=x^3-9*x-12;
      x0:read("enter the approximate value x0:")$
      c:newton(f(x),x,x0,0.00001);
      print("The required root is",c)$
      wxdraw2d(grid = true,
               point_type=filled_circle,
               explicit(f(x),x,-4,4),
               points([c,c],[0,0]))$
```

```
(%o3) f(x) := x^3 - 9 x - 12
      enter the approximate value x0: 0;
      (c) 3.522
      The required root is 3.522
```



Exercise: Write a maxima program to find a real root of the following equations using Newton Raphson Method.

3.1 $x^3 - 2x - 5 = 0$ (Ans: 2. 095)

3.2 $x^3 + x^2 + 3x + 4 = 0$ (Ans: -1. 222)

3.3 $x^3 - 37 = 0$ (Ans: 3. 332)

3.4 $\sin x = x - 2$ (Ans: 2. 554)

3.5 $x \sin x + \cos x = 0$ (Ans: 2.798)

Program-04: Solving system of equations by Gauss elimination method.

Example: Maxima program to solve

$$2x + y + z = 10$$

$$3x + 2y + 3z = 18$$

$$x + 4y + 9z = 16$$

by Gauss elimination method.

```
(%i10) kill(all)$
e1:2*x+y+z=10;
e2:3*x+2*y+3*z=18;
e3:x+4*y+9*z=16;
A:coefmatrix([e1,e2,e3],[x,y,z]);
B:addcol(A,[rhs(e1),rhs(e2),rhs(e3)])$
M:triangularize(B);
r1:M[1,1]*x+M[1,2]*y+M[1,3]*z=M[1,4];
r2:M[2,1]*x+M[2,2]*y+M[2,3]*z=M[2,4];
r3:M[3,1]*x+M[3,2]*y+M[3,3]*z=M[3,4];
linsolve([r1,r2,r3],[x,y,z]);

(e1)      z + y + 2 x = 10
(e2)      3 z + 2 y + 3 x = 18
(e3)      9 z + 4 y + x = 16

(A)      [ 2  1  1 ]
          [ 3  2  3 ]
          [ 1  4  9 ]

(M)      [ 2  1  1  10 ]
          [ 0  1  3   6 ]
          [ 0  0 -2 -10 ]

(r1)      z + y + 2 x = 10
(r2)      3 z + y = 6
(r3)      -2 z = -10
(%o10)    [x = 7 , y = -9 , z = 5 ]
```

Exercise: Write a maxima program to solve the following by Gauss elimination method.

4.1 $x + 4y - z = -5$ $\left[x = \frac{117}{71}, y = -\frac{81}{71}, z = \frac{148}{71} \right]$

$$x + y - 6z = -12$$

$$3x - y - z = 4$$

4.2 $x + y + z = 6$ $[x = 3, y = 1, z = 2]$

$$3x + 3y + 4z = 20$$

$$2x + y + 3z = 13$$

4.3 $2x + 2y + z = 12$ $\left[x = -\frac{51}{4}, y = \frac{115}{8}, z = \frac{35}{4} \right]$

$$3x + 2y + 2z = 8$$

$$5x + 10y - 8z = 10$$

4.4 $5x_1 + x_2 + x_3 + x_4 = 4$ $[x_1 = 1, x_2 = 2, x_3 = -1, x_4 = -2]$

$$x_1 + 7x_2 + x_3 + x_4 = 12$$

$$x_1 + x_2 + 6x_3 + x_4 = -5$$

$$x_1 + x_2 + x_3 + 4x_4 = -6$$

4.5 $x_1 - x_2 + x_3 + x_4 = 6$ $\left[x_1 = \frac{17}{3}, x_2 = 6, x_3 = \frac{5}{3}, x_4 = \frac{14}{3} \right]$

$$2x_1 - x_3 - x_4 = 5$$

$$2x_1 - 2x_2 + x_4 = 4$$

$$x_2 + x_3 - x_4 = 3$$

Program-05: Solving system of equations by Jacobi iteration method.

Example: Maxima program to solve

$$10x + y + z = 12$$

$$2x + 10y + z = 13$$

$$2x + 2y + 10z = 14$$

by Jacobi iteration method.

```
→ kill(all)$
e1:10*x+y+z=12;
e2:2*x+10*y+z=13;
e3:2*x+2*y+10*z=14;
a:ev(x,solve(e1,x))$
b:ev(y,solve(e2,y))$
c:ev(z,solve(e3,z))$
print(x=a)$
print(y=b)$
print(z=c)$
x[0]:read("enter the initial approximation x0=")$
y[0]:read("enter the initial approximation y0=")$
z[0]:read("enter the initial approximation z0=")$
for i:1 unless flag=0 do (
  x[i]:ev(a,y=y[i-1],z=z[i-1]),
  y[i]:ev(b,x=x[i-1],z=z[i-1]),
  z[i]:ev(c,x=x[i-1],y=y[i-1]),
  print(i,"approximation"),
  print("-----"),
  print('x[i]=float(x[i])),
  print('y[i]=float(y[i])),
  print('z[i]=float(z[i])),
  print("-----"),
  if abs(x[i]-x[i-1])<0.01 then
    (flag:0,
     print("The solution is",x=round(x[i]),",",y=round(y[i]),",",z=round(z[i])))$
```

$$\begin{aligned} \text{(e1)} \quad & z + y + 10x = 12 \\ \text{(e2)} \quad & z + 10y + 2x = 13 \\ \text{(e3)} \quad & 10z + 2y + 2x = 14 \end{aligned}$$

$$x = -\frac{z + y - 12}{10}$$

$$y = -\frac{z + 2x - 13}{10}$$

$$z = -\frac{y + x - 7}{5}$$

enter the initial approximation $x_0 = 0$;

ARRSTORE: use_fast_arrays=false; allocate a new property hash table for \$X

enter the initial approximation $y_0 = 0$;

ARRSTORE: use_fast_arrays=false; allocate a new property hash table for \$Y

enter the initial approximation $z_0 = 0$;

ARRSTORE: use_fast_arrays=false; allocate a new property hash table for \$Z

1 approximation

$$x_1 = 1.2$$

$$y_1 = 1.3$$

$$z_1 = 1.4$$

2 approximation

$$x_2 = 0.93$$

$$y_2 = 0.92$$

$$z_2 = 0.9$$

3 approximation

$$x_3 = 1.018$$

$$y_3 = 1.024$$

$$z_3 = 1.03$$

4 approximation

$$x_4 = 0.9946$$

$$y_4 = 0.9934$$

$$z_4 = 0.9916$$

5 approximation

$$x_5 = 1.0015$$

$$y_5 = 1.00192$$

$$z_5 = 1.0024$$

The solution is $x = 1$, $y = 1$, $z = 1$

Exercise: Write a maxima program to solve the following by Jacobi iteration method.

5.1 $20x + y - 2z = 17$ $[x = 1, y = -1, z = 1]$

$$3x + 20y - z = -18$$

$$2x - 3y + 20z = 25$$

5.2 $5x + 2y + z = 12$ $[x = 1, y = 2, z = 3]$

$$x + 4y + 2z = 15$$

$$x + 2y + 5z = 20$$

5.3 $5x - y = 9$ $[x = 2, y = 1, z = -1]$

$$-x + 5y - z = 4$$

$$y - 5z = 6$$

5.4 $5x - y + z = 10$ $[x = 2.556, y = 1.722, z = -1.055]$

$$x + 2y = 6$$

$$x + y + 5z = -1$$

with $(2, 3, 0)$ as initial approximation to the solution.

5.5 $5x - y + 3z = 10$ $[x = 4, y = 1, z = -3]$

$$3x + 6y = 18$$

$$x + y + 5z = -10$$

with $(3, 0, -2)$ as initial approximation to the solution.

Note: To get decimal values in the solution use *float* command instead of *round* command in the program.

Program-06: Solving system of equations by Gauss - Seidel iterative method.

Example: Maxima program to solve

$$27x + 6y - z = 85$$

$$6x + 15y + 2z = 72$$

$$x + y + 54z = 110$$

by Jacobi iteration method.

```
→ kill(all)$
fpprintprec:5$
e1:27*x+6*y-z=85;
e2:6*x+15*y+2*z=72;
e3:x+y+54*z=110;
a:ev(x,solve(e1,x))$
b:ev(y,solve(e2,y))$
c:ev(z,solve(e3,z))$
print(x=a)$
print(y=b)$
print(z=c)$
x[0]:read("enter the initial approximation x0=")$
y[0]:read("enter the initial approximation y0=")$
z[0]:read("enter the initial approximation z0=")$
for i:1 unless flag=0 do (
x[i]:ev(a,y=y[i-1],z=z[i-1]),
  y[i]:ev(b,x=x[i],z=z[i-1]),
  z[i]:ev(c,x=x[i],y=y[i]),
  print(i,"approximation"),
  print("-----"),
print('x[i]=float(x[i])),
  print('y[i]=float(y[i])),
  print('z[i]=float(z[i])),
  print("-----"),
if abs(x[i]-x[i-1])<0.01 then
  (flag:0,
  print("The solution is",x=round(x[i]),",",y=round(y[i]),",",z=round(z[i]))))$
```

(e1) $-z + 6y + 27x = 85$

(e2) $2z + 15y + 6x = 72$

(e3) $54z + y + x = 110$

$$x = \frac{z - 6y + 85}{27}$$

$$y = -\frac{2z + 6x - 72}{15}$$

$$z = -\frac{y + x - 110}{54}$$

enter the initial approximation $x_0 = 0$;

ARRSTORE: use_fast_arrays=false; allocate a new property hash table for \$X

enter the initial approximation $y_0 = 0$;

ARRSTORE: use_fast_arrays=false; allocate a new property hash table for \$Y

enter the initial approximation $z_0 = 0$;

ARRSTORE: use_fast_arrays=false; allocate a new property hash table for \$Z

1 approximation

$$x_1 = 3.1481$$

$$y_1 = 3.5407$$

$$z_1 = 1.9132$$

2 approximation

$$x_2 = 2.4322$$

$$y_2 = 3.572$$

$$z_2 = 1.9258$$

3 approximation

$$x_3 = 2.4257$$

$$y_3 = 3.5729$$

$$z_3 = 1.926$$

The solution is $x = 2$, $y = 4$, $z = 2$

Exercise: Write a maxima program to solve the following by Gauss - Seidel iterative method.

$$\begin{array}{ll} \mathbf{6.1} & \mathbf{10x + y + z = 12} \qquad \qquad \qquad [\mathbf{x = 1, y = 1, z = 1}] \\ & \mathbf{x + 10y + z = 12} \\ & \mathbf{x + y + 10z = 12} \end{array}$$

$$\begin{array}{ll} \mathbf{6.2} & \mathbf{5x + 2y + z = 12} \qquad \qquad \qquad [\mathbf{x = 1, y = 2, z = 3}] \\ & \mathbf{x + 4y + 2z = 15} \\ & \mathbf{x + 2y + 5z = 20} \end{array}$$

6.3 **$28x + 4y - z = 32$** **$[x = 0.9933, y = 1.507, z = 1.849]$**
 $2x + 17y + 4z = 35$
 $x + 3y + 10z = 24$

$$\begin{array}{ll} \mathbf{6.4} & \mathbf{x + 17y - 2z = 48} \qquad \mathbf{[x = 1.675, y = 2.862, z = 1.163]} \\ & \mathbf{2x + 2y + 18z = 30} \\ & \mathbf{30x - 2y + 3z = 48} \end{array}$$

6.5 **$9x - y + 2z = 9$** **$[x = 0.917, y = 1.647, z = 1.195]$**
 $x + 10y - 2z = 15$
 $2x - 2y - 13z = -17$

Note: To get decimal values in the solution use *float* command instead of *round* command in the program.

Program-07: To find the value of $f(x)$ at any point using Newton-Gregory forward interpolation formula.

If $f(x_0) = y_0, f(x_1) = y_1, \dots, f(x_n) = y_n$ be a set of values of an unknown function $y = f(x)$ corresponding to the values $x: x_0, x_1, x_3, \dots, x_n$ at equal intervals then the **Newton-Gregory forward interpolation formula** is given by

$$f(x + nh) = f(x) + \frac{n}{1!} \Delta f(x) + \frac{n(n-1)}{2!} \Delta^2 f(x) + \frac{n(n-1)(n-2)}{3!} \Delta^3 f(x) + \dots + \Delta^n f(x)$$

or

$$y_x = y_0 + \frac{n}{1!} \Delta y_0 + \frac{n(n-1)}{2!} \Delta^2 y_0 + \frac{n(n-1)(n-2)}{3!} \Delta^3 y_0 + \dots + \Delta^n y_0$$

Example: Maxima program to find $f(1.4)$ using Newton-Gregory forward interpolation formula given that.

x	1	2	3	4	5
$f(x)$	10	26	58	112	194

```
(%i14) kill(all)$
x:[1,2,3,4,5];
y:[10,26,58,112,194];
n:length(x)$
fpprintprec:6$
print("First forward difference is")$
for i:2 thru n do
(z[i]:=y[i]-y[i-1],
print(z[i]))$
print("Second forward difference is")$
for j:3 thru n do
(u[j]:=z[j]-z[j-1],
print(u[j]))$
print("Third forward difference is")$
for k:4 thru n do
(w[k]:=u[k]-u[k-1],
print(w[k]))$
X:1.4;
h:x[2]-x[1]$
p:float(X-x[1])/h;
y:y[1]+binomial(p,1)·z[2]+binomial(p,2)·u[3]+binomial(p,3)·w[4];
```

```

(x)      [1,2,3,4,5]
(y)      [10,26,58,112,194]
First forward difference is
16
32
54
82
Second forward difference is
16
22
28
Third forward difference is
6
6
(x)      1.4
(p)      0.4
(y)      14.864

```

Exercise:

7.1 Use Newton-Gregory interpolation formula to find y at $x = 2.5$ given

x	1	2	3	4	5	6
$f(x)$	1	8	27	64	125	216

7.2 Use Newton-Gregory interpolation formula to find y $f(0.33)$ given

x	0.3	0.4	0.5	0.6
$f(x)$	0.6179	0.6554	0.6915	0.7257

7.3 Use Newton-Gregory interpolation formula to estimate the population for the year 1985 from the table

Year	1970	1980	1990	2000	2010	2020
Population in crores	55.75	69.68	87.05	105.96	124.06	139.64

7.4 From the following data find the number of students who obtained less than 45 marks using Newton-Gregory interpolation formula

Marks	30-40	40-50	50-60	60-70	70-80
Number of students	31	42	51	35	31

Program-08: Lagrange's Interpolation.

If $f(x_0) = y_0, f(x_1) = y_1, \dots, f(x_n) = y_n$ be a set of values of an unknown function $y = f(x)$ corresponding to the values $x: x_0, x_1, x_2, \dots, x_n$ not necessarily at equal intervals then the **Lagrange's interpolation formula** is given by

$$y = f(x) = \frac{(x-x_1)(x-x_2)\dots(x-x_n)}{(x_0-x_1)(x_0-x_2)\dots(x_0-x_n)} y_0 + \frac{(x-x_0)(x-x_2)\dots(x-x_n)}{(x_1-x_0)(x_1-x_2)\dots(x_1-x_n)} y_1 \\ + \dots + \frac{(x-x_0)(x-x_1)\dots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\dots(x_n-x_{n-1})} y_n$$

Example: Maxima program to find $f(4)$ using Lagrange's interpolation formula given that.

x	0	2	3	6
$f(x)$	-4	2	14	158

```
→ kill(all)$
load(interpol)$
powerdisp:true$
a:[[x0,y0],[x1,y1],[x2,y2],[x3,y3]]$
disp("Lagrange's interpolation formula for four values is given by ", "y=f(x)=", lagrange(a))$
p:[[0,-4],[2,2],[3,14],[6,158]];
lagrange(p)$
f(x):="%;
print("Lagrange's interpolation polynomial is", ratsimp(%))$
'f(4)=f(4);
q:[[4,f(4)]]$
wxdraw2d(
  color    = red,
  key      = "Lagrange polynomial",
  explicit(f(x), x, 0, 10),
  yrange=[-50, 200],
  color    = blue,
  point_type=filled_circle,
  key      = "Points",
  points(p),
  color    = black,
  key      = "Required Point",
  points(q))$
```

Lagrange's interpolation formula for four values is given by

$$y=f(x)=\frac{(x-x_1)(x-x_2)(x-x_3)y_0}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}+\frac{(x-x_0)(x-x_2)(x-x_3)y_1}{(-x_0+x_1)(x_1-x_2)(x_1-x_3)}+\frac{(x-x_0)(x-x_1)(x-x_3)y_2}{(-x_0+x_2)(-x_1+x_2)(x_2-x_3)}+\frac{(x-x_0)(x-x_1)(x-x_2)y_3}{(-x_0+x_3)(-x_1+x_3)(-x_2+x_3)}$$

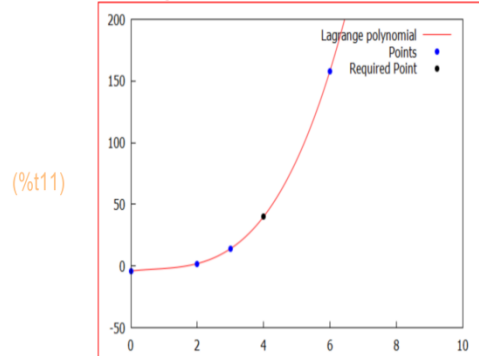
(p) **[[0,-4],[2,2],[3,14],[6,158]]**

(%o7) $f(x) := \frac{(-6+x)(-3+x)(-2+x)}{9} + \frac{(-6+x)(-3+x)x}{4} - \frac{14(-6+x)(-2+x)x}{9} + \frac{79(-3+x)(-2+x)x}{36}$

Lagrange's interpolation polynomial is $f(x) := -4 + 3x - 2x^2 + x^3$

(%o9) $f(4) = 40$

0 errors, 0 warnings



Exercise:

8.1 Use Lagrange's interpolation formula to find y at x= 10 given

x	5	6	9	11
$f(x)$	12	13	14	16

8.2 Use Lagrange's interpolation formula to find y at x= 1.6 given

x	1.2	2.0	2.5	3.0
$f(x)$	1.36	0.58	0.34	0.20

8.3 Use Lagrange's interpolation formula to find f(9) given

x	5	7	11	13	17
$f(x)$	150	392	1452	2366	5202

8.4 The following table gives the normal weights of babies during first eight months of life

Age (in months)	0	2	5	8	10
Weight (in Kg)	2.5	4.4	6.1	7.0	7.5

Estimate the weight of the baby at the age of seven months using Lagrange's interpolation formula.

Program-09: To evaluate definite integrals using Simpson's one-third rule.

Let $I = \int_a^b f(x)dx$, where $y = f(x)$ takes the values $y_0, y_1, y_2, \dots, y_n$ for $x = x_0, x_1, x_2, \dots, x_n$.

Divide the interval (a, b) into even number of equal sub-intervals of width h. Simpson's one-third rule is as follows

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{3}[(y_0 + y_n) + 4(y_1 + y_3 + \dots + y_{n-1}) + 2(y_2 + y_4 + \dots + y_{n-2})]$$

Example: Write a maxima program to evaluate $\int_0^6 \frac{1}{1+x} dx$ with n = 6 using Simpson's one-third rule.

```
(%i10) kill(all)$
      fpprintprec:4$
      f(x):=1/(1+x);
      x0:read("enter the lower limit x0:")$
      xn:read("enter the upper limit xn:")$
      n:read("enter the number of intervals n:")$
      h:(xn-x0)/n$
      sum1:0$
      sum2:0$
      for i:1 thru n-1 do
      if mod(i,2)=0 then
      sum1:sum1+f(x0+i*h)
      else
      sum2:sum2+f(x0+i*h)$
      'integrate(f(x),x,x0,xn)=float(h/3*(f(x0)+f(xn)+4*sum2+2*sum1));
(%o2)  f(x):= 1
        1+x
enter the lower limit x0: 0;
enter the upper limit xn: 6;
enter the number of intervals n: 6;

(%o10)  $\int_0^6 \frac{1}{x+1} dx = 1.959$ 
```

Exercise:

Write a maxima program to evaluate the following using Simpson's one-third rule.

9.1 $\int_0^6 \frac{1}{1+x^2} dx$, $n=6$ (Ans:1.366)

9.2 $\int_0^6 \frac{x}{1+x^2} dx$, $n=6$ (Ans: 1.801)

9.3 $\int_0^1 \frac{x}{1+x^4} dx$, $n=4$ (Ans:0.393)

9.4 $\int_0^{1.2} \frac{1}{1+x} dx$, $n=6$ (Ans:0.7885)

9.5 $\int_1^2 \frac{1}{x} dx$, $n=4$ (Ans: 0.6933)

Program-10: To evaluate definite integrals using Simpson's three-eight rule.

Let $I = \int_a^b f(x)dx$, where $y = f(x)$ takes the values $y_0, y_1, y_2, \dots, y_n$ for $x = x_0, x_1, x_2, \dots, x_n$.

Divide the interval (a,b) into a number which is multiple of 3 sub-intervals .

Simpson's Three-eight rule is as follows

$$\int_{x_0}^{x_n} f(x)dx = \frac{3}{8}h[(y_0 + y_n) + 3(y_1 + y_2 + y_4 + y_5 + \dots + y_{n-4} + y_{n-2} + y_{n-1}) + 2(y_3 + y_6 + \dots + y_{n-3})]$$

Example: Write a maxima program to evaluate $\int_0^3 \frac{1}{(1+x)^2} dx$ with $n = 6$ using Simpson's three-eight rule.

```
(%i10) kill(all)$
fpprintprec:4$
f(x):=1/(1+x)^2;
x0:read("enter the lower limit x0:")$
xn:read("enter the upper limit xn:")$
n:read("enter the number of intervals n:")$
h:(xn-x0)/n$
sum1:0$
sum2:0$
for i:1 thru n-1 do
if mod(i,3)=0 then
sum1:sum1+f(x0+i*h)
else
sum2:sum2+f(x0+i*h)$
'integrate(f(x),x,x0,xn)=float(3*h/8*(f(x0)+f(xn)+3*sum2+2*sum1));
```

```
(%o2) f(x) := \frac{1}{(1+x)^2}

enter the lower limit x0: 0;
enter the upper limit xn: 3;
enter the number of intervals n: 6;
```

```
(%o10)  $\int_0^3 \frac{1}{(x+1)^2} dx = 0.7583$ 
```

Exercise: Write a maxima program to evaluate the following using Simpson's three-eighth rule.

10.1 $\int_0^{0.3} (2x + x^2)^{1/2} dx$, **n=6** **(Ans:0.1602)**

10.2 $\int_0^3 \frac{1}{x^2+4} dx$, **n=6** **(Ans: 0.4913)**

10.3 $\int_1^4 e^{1/x} dx$, **n=6** **(Ans: 4.866)**

10.4 $\int_4^{5.2} \log x \, dx$, **n=6** **(Ans: 1.828)**

10.5 $\int_0^{0.6} (1 - 3x^4)^{1/2} dx$, **n=6** **(Ans: 0.5751)**