# Mata Sundri College for Women

**University of Delhi**

**PRACTICAL FILE**

**of**

**NUMERICAL METHODS**

# INDEX

# PROGRAM # 1
# “BISECTION METHOD”

## Aim :

To perform the iterations of Bisection method for the functions f1(x)= x^3+2*x^2-3*x-1, f2(x)= x^3+2*x^2-3*x-3 and f3(x)= sin [x] on the intervals [1,2], [1,2] and [3,4] respectively

## Programming:

```
Bisection[a0_, b0_, m_] :=
 Module[{a = N[a0], b = N[b0]}, c = (a + b)/2;
  k = 0;
  While[k < m,
   If[ Sign[f[b]] == Sign[f[c]], b = c, a = c;];
   c = (a + b)/2;
   k = k + 1;];
  Print["c= ", NumberForm[c, 16]];
  Print["f[c]= ", NumberForm[f[c], 16]];];
```

**(i)** f[x_]:=x^3+2*x^2-3*x-1

```
Bisection[1, 2, 30];
```

## Output :

c= 1.198691243771464
f[c]= 1.559712359266996*10^(-9)

**(ii)** f[x_] := x^3 + x^2 - 3*x – 3

```
Bisection[1, 2, 30];
```

## Output :

c= 1.732050807680935
f[c]= 1.060522336615577*10^(-9)

**(iii)** f[x_] := Sin[x]

```
Bisection[3, 4, 20];
```

## Output :

c= 3.14159250259399
f[c]= 1.50995799097837*10^(-7)

# PROGRAM # 2
# "BISECTION METHOD WITH CONDITION OF ABSOLUTE CONVERGENCE"

## Aim:

To perform the iterations of Bisection method for the functions f1(x)= x^3+2*x^2-3*x-1, f2(x)= x^3+2*x^2-3*x-3 and f3(x)= sin[x] on the intervals [1,2], [1,2] and [3,4] respectively within an absolute convergence of10^(-7)

## Programming:

```
Bisection[a0_, b0_, m_] :=
 Module[{a = N[a0], b = N[b0]}, c = (a + b)/2;
  k = 0;
  While[k < m && ((b - a)/2) > 0.0000001,
   If[ Sign[f[b]] == Sign[f[c]], b = c, a = c;];
   c = (a + b)/2;
   k = k + 1;];
  Print["c= ", NumberForm[c, 16]];
  Print["f[c]= ", NumberForm[f[c], 16]];];
```

**(i)** f[x_] := x^3 + 2*x^2 - 3*x – 1

Bisection[1, 2, 30];

## Output :

c= 1.198691189289093
f[c]= -3.310740535056311*10^(-7)

**(ii)** f[x_] := x^3 + x^2 - 3*x – 3

Bisection[1, 2, 30];

## Output :

c= 1.732050807680935
f[c]= 1.060522336615577*10^(-9)

**(iii)** f[x_] := Sin[x]

Bisection[3, 4, 20];

## Output :

c= 3.141592502593994
f[c]= 1.50995799097837*10^(-7)

# PROGRAM # 3
# “REGULA FALSI METHOD”

## Aim:
To perform the iterations of Regula Falsi method for the functions f1(x)= x^3 + x^2 - 3*x – 3, f2(x)= 1 - Log[x] and f3(x)= x^6-3 on the intervals [1,2], [2,3] and [1,2] respectively

## Programming:
```
RegulaFalsi[a0_, b0_, m_] :=
 Module[{},
  a = N[a0];
  b = N[b0];
  c = (a*f[b] - b*f[a])/(f[b] - f[a]);
  k = 0;
  While[k < m,
   If[Sign[f[b]] == Sign[f[c]],
    b = c, a = c;];
   c = (a*f[b] - b*f[a])/(f[b] - f[a]);
   k = k + 1;];
  Print["c= ", NumberForm[c, 16]];
  Print["f[c]= ", NumberForm[f[c], 16]];]
```

**(i)** f[x_] := x^3 + x^2 - 3*x – 3

RegulaFalsi[1, 2, 30];

## Output :
c= 1.732050807568877
f[c]= -1.77635683940025*10^(-15)

**(ii)** f[x_] := 1 - Log[x]

RegulaFalsi[2, 3, 10];

## Output :
c= 2.718281828561478
f[c]= -3.768274581261721*10^(-11)

**(iii)** f[x_] := x^6 – 3

RegulaFalsi[1, 2, 15];

## Output :
c= 1.192944816338086
f[c]= -0.117813190137853

# PROGRAM # 4
# “REGULA FALSI METHOD WITH CONDITION OF ABSOLUTE CONVERGENCE”

## Aim:

To perform the iterations of Regula Falsi method for the functions f1(x)= x^3+2*x^2-3*x-1, f2(x)= x^5 + 2*x – 1 and f3(x)= e^(-x) – x on the intervals [1,2], [0,1] and [0,1] respectively within an absolute convergence of10^(-12)

## Programming:

```
RegulaFalsi[a0_, b0_, m_] :=
 Module[{},
  a = N[a0];
  b = N[b0];
  c = (a*f[b] - b*f[a])/(f[b] - f[a]);
  k = 0;
  While[(k < m && Abs[f[c]] > 0.000000000001),
   If[Sign[f[b]] == Sign[f[c]],
    b = c, a = c;];
   c = (a*f[b] - b*f[a])/(f[b] - f[a]);
   k = k + 1;];
  Print["c= ", NumberForm[c, 16]];
  Print["f[c]= ", NumberForm[f[c], 16]];]
```

**(i)** f[x_] := x^3 + 2*x^2 - 3*x – 1

RegulaFalsi[1, 2, 40];

## Output :

c= 1.19869124351587
f[c]= -7.780442956573097*10^(-13)

**(ii)** f[x_] := x^5 + 2*x – 1

RegulaFalsi[0, 1, 30];

## Output :

c= 0.73908513321505
f[c]= 1.849631559025511*10^(-13)

**(iii)** f[x_] := e^(-x) – x

RegulaFalsi[0, 1, 30];

## Output :

c= 0.5671432904099458
f[c]= -2.537969834293108*10^(-13)

# PROGRAM # 5
# "NEWTON RAPHSON'S METHOD"

## Aim:
To perform the iterations of Newton Raphson's method for the functions f1(x)= x^3+2*x^2-3*x-1, f2(x)= Cos[x] – x and f3(x)= e^(-x) – x on the intervals [1,2], [0,1] and [0,1] respectively

## Programming:
```
NewtonRaphson[x0_, max_] :=
 Module[{},
  k = 0;
  p0 = N[x0];
  p1 = p0;
  While[k < max,
   p0 = p1;
   p1 = p0 - f[p0]/f'[p0];
   k = k + 1;];
  Print["p= ", NumberForm[p1, 16]];
  Print["f[p]= ", NumberForm[f[p1], 16]];];
```

(i) f[x_] := x^3 + 2*x^2 - 3*x – 1

NewtonRaphson[2, 5];

## Output :
p= 1.19869124352843
f[p]= 7.59046159259924*10^(-11)

(ii) f[x_] := Cos[x] – x

NewtonRaphson[1, 3];

## Output :
p= 0.739085133385284
f[p]= -2.847205804457076*10^(-10)

(iii) f[x_] := e^(-x) – x

NewtonRaphson[1, 3];

## Output :
p= 0.567143285989123
f[p]= 6.927808993140161*10^(-9)

# PROGRAM # 6
# "NEWTON RAPHSON'S METHOD WITH CONDITION OF ABSOLUTE CONVERGENCE"

## Aim:

To perform the iterations of Newton Raphson's method for the functions f1(x)= x^3+2*x^2-3*x-1, f2(x)= Cos[x] – x and f3(x)= e^(-x) – x on the intervals [1,2], [0,1] and [0,1] respectively within an absolute convergence of 10^(-8)

## Programming:

```
NewtonRaphson[x0_, max_] :=
 Module[{},
  k = 0;
  p0 = N[x0];
  p1 = p0;
  While[(k < max && Abs[f[p1]] > 0.00000001),
   p0 = p1;
   p1 = p0 - f[p0]/f'[p0];
   k = k + 1;];
  Print["p= ", NumberForm[p1, 16]];
  Print["f[p]= ", NumberForm[f[p1], 16]];];
```

**(i)** f[x_] := x^3 + 2*x^2 - 3*x – 1

```
NewtonRaphson[2, 13];
```

## Output :

p= 1.19869124352843
f[p]= 7.59046159259924*10^(-11)

**(ii)** f[x_] := Cos[x] – x

```
NewtonRaphson[1, 3];
```

## Output :

p= 0.739085133385284
f[p]= -2.847205804457076*10^(-10)

**(iii)** f[x_] := e^(-x) – x

```
NewtonRaphson[1, 3];
```

## Output :

p= 0.567143285989123
f[p]= 6.927808993140161*10^(-9)

# PROGRAM # 7
# "SECANT METHOD"

## Aim:

To perform the iterations of Secant Method for the functions f1(x)= x^3 - 2*x – 5, f2(x)= Sin[x] and f3(x)= Cos[x] – x on the intervals [2,3], [3,4] and [0,1] respectively

## Programming:

```
SecantMethod[x0_, x1_, max_] :=
 Module[{},
  k = 1;
  p0 = N[x0];
  p1 = N[x1];
  p2 = p1;
  p1 = p0;
  While[k < max,
   p0 = p1;
   p1 = p2;
   p2 = p1 - (f[p1] (p1 - p0)/(f[p1] - f[p0]));
   k = k + 1;];
  Print["p", k, "=", NumberForm[p2, 11]];
  Print["f[p", k, "]=", NumberForm[f[p2], 11]];];
```

**(i)** f[x_] := x^3 - 2*x – 5

SecantMethod[3, 2, 4]

## Output :

p4=0.74213345706
f[p4]=-0.0051051421631

**(ii)** f[x_] := Sin[x]

SecantMethod[3, 4, 5]

## Output :

p5=3.1415926536
f[p5]=5.6521794331*10^(-14)

**(iii)** f[x_] := Cos[x] – x

SecantMethod[0, 1, 6]

## Output :

p6=0.73908513322
f[p6]=2.6678659282*10^(-13)

# PROGRAM # 8
# "SECANT METHOD WITH CONDITION OF ABSOLUTE CONVERGENCE"

## Aim:

To perform the iterations of Secant method for the functions f1(x)= x^3 - 2*x – 5 , f2(x)= Sin[x] and f3(x)= Cos[x] – x on the intervals [2,3], [3,4] and [0,1] respectively within an absolute convergence of 5*10^(-7)

## Programming:

```
SecantMethod[x0_, x1_, max_] :=
 Module[{},
  k = 1;
  p0 = N[x0];
  p1 = N[x1];
  p2 = p1;
  p1 = p0;
  While[(k < max && Abs[f[p2]] > 5*10^(-7)),
   p0 = p1;
   p1 = p2;
   p2 = p1 - (f[p1] (p1 - p0)/(f[p1] - f[p0]));
   k = k + 1;];
  Print["p", k, "=", NumberForm[p2, 11]];
  Print["f[p", k, "]=", NumberForm[f[p2], 11]];];
```

**(i)** f[x_] := x^3 - 2*x – 5

SecantMethod[3, 2, 50]

## Output :

p6=2.0945514815
f[p6]=1.1884715434*10^(-11)

**(ii)** f[x_] := Sin[x]

SecantMethod[3, 4, 20]

## Output :

p4=3.141592728
f[p4]=-7.4395063765*10^(-8)

**(iii)** f[x_] := Cos[x] – x

SecantMethod[0, 1, 30]

## Output :

p5=0.73908511213
f[p5]=3.5292622824*10^(-8)

## PROGRAM # 9
## "LU DECOMPOSITION METHOD"

### Aim:
To perform LU Decomposition on a matrix

### Programming:
$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & -4 & 3 \\ 3 & 2 & 2 \end{pmatrix}$$

LUDecomposition[A]

### Output :
{{2, 1, 1}, {1, -4, 3}, {3, 2, 2}}

{{{1, -4, 3}, {2, 9, -5}, {3, 14/9, 7/9}}, {2, 1, 3}, 1}

# PROGRAM # 10
# “GAUSS JACOBI METHOD”

## Aim:

To solve the following system of linear equations by using Gauss-Jacobi Mehod : 4*x1-x2 =2
-x1+4*x2-x3=4
-x2+4*x3=10

## Programming:

```
Jacobi[A0_, B0_, X0_, max_] :=
 Module[{A = N[A0], B = N[B0], i, j, k = 0, n = Length[X0], X = X0,
   Xold = X0},
  Print["X", 0, "=", X];
  While[k < max,
   For[i = 1, i <= n, i = i + 1,
    X[[i]] = (B[[i]] + A[[i, i]]*Xold[[i]] - ∑_{j=1}^{n} A[[i, j]]*Xold[[j]])/A[[i, i]]];
   Print["X", k + 1, "=", X];
   Xold = X;
   k = k + 1;];];
```

A0 = $\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$;

B0 = $\begin{pmatrix} 2 \\ 4 \\ 10 \end{pmatrix}$;

X0 = $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$;

Jacobi[A0, B0, X0, 10];

## Output :

X0={{0},{0},{0}}
X1={{0.5},{1.},{2.5}}
X2={{0.75},{1.75},{2.75}}
X3={{0.9375},{1.875},{2.9375}}
X4={{0.96875},{1.96875},{2.96875}}
X5={{0.9921875},{1.984375},{2.9921875}}
X6={{0.99609375},{1.99609375},{2.99609375}}
X7={{0.9990234375},{1.998046875},{2.999023438}}
X8={{0.9995117188},{1.999511719},{2.999511719}}
X9={{0.9998779297},{1.999755859},{2.99987793}}
X10={{0.9999389648},{1.999938965},{2.999938965}}

# PROGRAM # 11
# "GAUSS JACOBI METHOD WITH CONDITION OF ABSOLUTE CONVERGENCE"

## Aim :

To solve the following system of linear equations by using Gauss-Jacobi Mehod within an absolute tolerance of 5*10(-6): 4*x1-x2 =2
-x1+4*x2-x3=4
-x2+4*x3=10

## Programming:

```
Jacobi[A0_, B0_, X0_, max_] :=
 Module[{A = N[A0], B = N[B0], i, j, k = 0, n = Length[X0], X = X0,
   Xold = X0},
  Print["X", 0, "=", X];
  While[k < max,
   For[i = 1, i <= n, i = i + 1,
    X[[i]] = (B[[i]] + A[[i, i]]*Xold[[i]] - \!\(\*UnderoverscriptBox[\(\[Sum]\), \(j = 1\), \(n\)]\(A[[i, j]]\)\)*Xold[[j]])/A[[i, i]]];
   Print["X", k + 1, "=", NumberForm[X, 10]];
   If[Max[Abs[X - Xold]] < 5*10^(-6),
    Print["Solution with convergence tolerance of 5*10^(-6) = ",
     NumberForm[X, 10]];
    Break[];,
    Xold = X;
    k = k + 1;];];]
```

$$A0 = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix};$$

$$B0 = \begin{pmatrix} 2 \\ 4 \\ 10 \end{pmatrix};$$

$$X0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

```
Jacobi[A0, B0, X0, 30];
```

## Output :

X0={{0},{0},{0}}
X1={{0.5},{1.},{2.5}}
X2={{0.75},{1.75},{2.75}}
X3={{0.9375},{1.875},{2.9375}}
X4={{0.96875},{1.96875},{2.96875}}
X5={{0.9921875},{1.984375},{2.9921875}}
X6={{0.99609375},{1.99609375},{2.99609375}}
X7={{0.9990234375},{1.998046875},{2.999023438}}
X8={{0.9995117188},{1.999511719},{2.999511719}}
X9={{0.9998779297},{1.999755859},{2.99987793}}
X10={{0.9999389648},{1.999938965},{2.999938965}}
X11={{0.9999847412},{1.999969482},{2.999984741}}
X12={{0.9999923706},{1.999992371},{2.999992371}}
X13={{0.9999980927},{1.999996185},{2.999998093}}
X14={{0.9999990463},{1.999999046},{2.999999046}}
Solution with convergence tolerance of 5*10^(-6) = {{0.9999990463},{1.999999046},{2.999999046}}

# PROGRAM # 12
# “GAUSS SEIDEL METHOD”

## Aim :

To solve the following system of linear equations by using Gauss-Seidel Mehod : 4*x1-x2 =2
-x1+4*x2-x3=4
-x2+4*x3=10

## Programming:

```
GaussSeidel[A0_, B0_, X0_, max_] :=
 Module[{A = N[A0], B = N[B0], i, j, k = 0, n = Length[X0], X = X0,
   Xold = X0},
  Print["X", 0, "=", X];
  While[k < max,
   For[i = 1, i <= n, i = i + 1,
    X[[i]] = (B[[i]] - Sum_{j=1}^{i-1} A[[i,j]] * X[[j]] - Sum_{j=i+1}^{n} A[[i,j]] * Xold[[j]])/A[[i, i]]];
   Print["X", k + 1, "=", NumberForm[X, 10]];
   Xold = X;
   k = k + 1;];];
```

$$X[[i]] = \left(B[[i]] - \sum_{j=1}^{i-1} A[[i,j]] * X[[j]] - \sum_{j=i+1}^{n} A[[i,j]] * Xold[[j]]\right)/A[[i,i]];$$

$$A0 = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix};$$

$$B0 = \begin{pmatrix} 2 \\ 4 \\ 10 \end{pmatrix};$$

$$X0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

```
GaussSeidel[A0, B0, X0, 10];
```

## Output :

```
X0={{0},{0},{0}}
X1={{0.5},{1.125},{2.78125}}
X2={{0.78125},{1.890625},{2.97265625}}
X3={{0.97265625},{1.986328125},{2.996582031}}
X4={{0.9965820313},{1.998291016},{2.999572754}}
X5={{0.9995727539},{1.999786377},{2.999946594}}
X6={{0.9999465942},{1.999973297},{2.999993324}}
X7={{0.9999933243},{1.999996662},{2.999999166}}
X8={{0.9999991655},{1.999999583},{2.999999896}}
X9={{0.9999998957},{1.999999948},{2.999999987}}
X10={{0.999999987},{1.999999993},{2.999999998}}
```

## PROGRAM # 13
## "GAUSS SEIDEL METHOD WITH CONDITION OF ABSOLUTE CONVERGENCE"

### Aim :

To solve the following system of linear equations by using Gauss-Seidel Mehod within an absolute tolerance of 5*10(-6):

```
4*x1-x2      =2
-x1+4*x2-x3=4
    -x2+4*x3=10
```

### Programming :

```
GaussSeidel[A0_, B0_, X0_, max_] :=
 Module[{A = N[A0], B = N[B0], i, j, k = 0, n = Length[X0], X = X0,
   Xold = X0},
  Print["X", 0, "=", X];
  While[k < max,
   For[i = 1, i <= n, i = i + 1,
    X[[i]] = (B[[i]] - Sum[A[[i,j]] * X[[j]], {j, 1, i-1}] - Sum[A[[i,j]] * Xold[[j]], {j, i+1, n}])/A[[i, i]];
   Print["X", k + 1, "=", NumberForm[X, 10]];
   If[Max[Abs[X - Xold]] < 5*10^(-6),
    Print["Solution with convergence tolerance of 5*10^(-6) = ",
     NumberForm[X, 10]];
    Break[];,
    Xold = X;
    k = k + 1;];];];]
```

where the summation line is printed as $X[[i]] = (B[[i]] - \sum_{j=1}^{i-1} A[[i,j]] * X[[j]] - \sum_{j=i+1}^{n} A[[i,j]] * \text{Xold}[[j]])/A[[i,i]];$

$$A0 = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix};$$

$$B0 = \begin{pmatrix} 2 \\ 4 \\ 10 \end{pmatrix};$$

$$X0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

```
GaussSeidel[A0, B0, X0, 30];
```

### Output :

```
X0={{0},{0},{0}}
X1={{0.5},{1.125},{2.78125}}
X2={{0.78125},{1.890625},{2.97265625}}
X3={{0.97265625},{1.986328125},{2.996582031}}
X4={{0.9965820313},{1.998291016},{2.999572754}}
X5={{0.9995727539},{1.999786377},{2.999946594}}
X6={{0.9999465942},{1.999973297},{2.999993324}}
X7={{0.9999933243},{1.999996662},{2.999999166}}
X8={{0.9999991655},{1.999999583},{2.999999896}}
X9={{0.9999998957},{1.999999948},{2.999999987}}
```

Solution with convergence tolerance of 5*10^(-6) = {{0.9999998957},{1.999999948},{2.999999987}

# PROGRAM # 14
# "LAGRANGE FORM OF INTERPOLATION"

## Aim :

To estimate the values of $e^{0.5}$, $e^{-0.7}$ and $e^{0.3}$ by constructing the Lagrange form of interpolating polynomial for f passing through (-1,$e^{-1}$), (0,1) and (1,$e^{1}$)

## Programming:

```
Lagrange1[x_, f_, y_] := Module[{},
  s = 0; m = Length[x]; p = 1;
  For[i = 1, i <= m, i = i + 1,
   For[j = 1, j <= m, j = j + 1,
    If[j ≠ i,
      p = p*(y - x[[j]])/(x[[i]] - x[[j]]); Continue;];];
   s = s + p*f[[i]]; p = 1;];
  Print["Function value at y=", s];
  Print["Absolute error=", Abs[s - e^y]];]

x = {-1, 0, 1};

f = {e^-1, 1, e^1};
```

(i) Lagrange1[x, f, 0.5]

## Output :

Function value at y=1.72337
Absolute error=0.0746495

(ii) Lagrange1[x, f, -0.7]

## Output :

Function value at y=0.443469
Absolute error=0.0531166

(iii) Lagrange1[x, f, 0.3]

## Output :

Function value at y=1.40144
Absolute error=0.0515788

# PROGRAM # 15
# "SIMPSON'S RULE"

## Aim :

To approximate the value of integrals $\int_1^2 x\,dx$, $\int_0^1 e^{-x} dx$ and $\int_0^1 1/(1+x^2)dx$ using Simpson 's Rule

## Programming:

```
Simpson1[a_, b_] := Module[{},
  I1 = ((b - a)/6)*(f1[a] + 4*f1[(a + b)/2] + f1[b]);
  Print["I1=", NumberForm[I1, 3]];];

f1[x_] = x;

Simpson1[1, 2]
```

## Output :

I1=3/2

```
f1[x_] = e^(-x);

Simpson1[0,1]
```

## Output :

I1=_1/6 (1+1/e+4/$\sqrt{e}$ )

```
f1[x_] = 1/(1 + x^2);

Simpson1[0, 1]
```

## Output :

I1=47/60

# PROGRAM # 16
# "SUM OF THE SERIES $\sum 1/N$"

## Aim :
To calculate the sum $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots\ldots + \frac{1}{n}$

## Programming : 1
```
s = 0;
n = 10;
For[k = 1, k < n + 1, k = k + 1, s = s + 1/k]
Print["Sum of the series=", N[s]];
```

## Output :
Sum of the series=2.92897

## Programming : 2
```
Sum[1/N, {N, 1, 10}]
```

## Output :
7381/2520

## Programming : 3
```
Seriessum[n_] := Module[{s = 0},
  For[i = 1, i < n + 1, i = i + 1, s = s + 1/i;
   Print["s", i, "=", NumberForm[s, 0]];];
  Print["Final sum of the series=", N[s]];]
```

(i) Seriessum[3]

## Output :
s1=1
s2=3/2
s3=11/6
Final sum of the series=1.83333

(ii) Seriessum[5]

## Output :
s1=1
s2=3/2
s3=11/6
s4=25/12
s5=137/60
Final sum of the series=2.28333

**(iii)** Seriessum[10]

## Output :

s1=1
s2=3/2
s3=11/6
s4=25/12
s5=137/60
s6=49/20
s7=363/140
s8=761/280
s9=7129/2520
s10=7381/2520
Final sum of the series=2.92897

# PROGRAM # 17
# "SUM OF THE NATURAL NUMBERS"

## Aim :
To calculate the sum of first 'n' natural numbers ,i.e., 1+2+3+....+n

## Programming : 1
```
s = 0;
n = 50;
For[k = 1, k < n + 1, k = k + 1, s = s + k]
Print["Sum of the series=", N[s]];
```

## Output :
Sum of the series=1275.

## Programming : 2
```
Sum[N, {N, 1, 20}]
```

## Output :
210

## Programming : 3
```
Seriessum[n_] := Module[{s = 0},
  For[i = 1, i < n + 1, i = i + 1, s = s + i;
   Print["s", i, "=", s];];
  Print["Final sum of the series=", s];]
```

(i) Seriessum[3]

## Output :
s1=1
s2=3
s3=6
Final sum of the series=6

(ii) Seriessum[5]

## Output :
s1=1
s2=3
s3=6
s4=10
s5=15
Final sum of the series=15

**(iii)** Seriessum[8]

## Output :

s1=1
s2=3
s3=6
s4=10
s5=15
s6=21
s7=28
s8=36
Final sum of the series=36

## PROGRAM # 18
## "SORTING INTEGERS IN THE ASCENDING ORDER"

### Aim :
To enter 'n' integers into a list and sort them in the ascending order

### Programming : 1
```
a = {1, 4, 19, 10, 34, 8};
n = Length[a];
For[i = 1, i < n, i = i + 1,
  For[j = i + 1, j < n + 1, j = j + 1,
    If[a[[i]] < a[[j]], Continue, temp = a[[i]]; a[[i]] = a[[j]];
      a[[j]] = temp;];];];

a
```

### Output :
{1, 4, 8, 10, 19, 34}

### Programming : 2
```
Asd1[a1_] := Module[{a = a1},
  n = Length[a];
  For[i = 1, i <= n, i = i + 1,
   For[j = i + 1, j < n + 1, j = j + 1,
     If[a[[i]] < a[[j]], Continue, temp = a[[i]]; a[[i]] = a[[j]];
       a[[j]] = temp;];];]; Print[a];]
```

(i) a1 = {23, 45.6, 78.5, 67};

Asd1[a1]

### Output :
{23,45.6,67,78.5}

(ii) a1 = {2, 5, 9, 11};

Asd1[a1]

### Output :
{2,5,9,11}

(iii) a1 = {34, 67, 89, 25};

Asd1[a1]

### Output :
{25,34,67,89}

## PROGRAM # 19
## "SORTING INTEGERS IN THE DESCENDING ORDER"

### Aim :
To enter 'n' integers into a list and sort them in the descending order

### Programming : 1
```
a = {5, 8, 35, 76, 2, 7, 90};
n = Length[a];
For[i = 1, i < n, i = i + 1,
  For[j = i + 1, j < n + 1, j = j + 1,
    If[a[[i]] > a[[j]], Continue, temp = a[[i]]; a[[i]] = a[[j]];
      a[[j]] = temp;];];];

 a
```

### Output :
{90, 76, 35, 8, 7, 5, 2}

### Programming : 2
```
Dsd1[a1_] := Module[{a = a1},
  n = Length[a];
  For[i = 1, i <= n, i = i + 1,
   For[j = i + 1, j < n + 1, j = j + 1,
     If[a[[i]] > a[[j]], Continue, temp = a[[i]]; a[[i]] = a[[j]];
      a[[j]] = temp;];];]; Print[a];]
```

(i) a1 = {23.5, 56.4, 8, 45};

Dsd1[a1]

### Output :
{56.4,45,23.5,8}

(ii) a1 = {2, 6, 7, 21};

Dsd1[a1]

### Output :
{21,7,6,2}

(iii) a1 = {234, 587, 908, 425};

Dsd1[a1]

### Output :
{908,587,425,234}

# PROGRAM # 20
# "ABSOLUTE VALUE OF AN INTEGER"

## Aim :
To find the absolute value of an integer

## Programming : 1
```
Abs[-2]
```

## Output :
2

## Programming : 2
```
x = {1, -4, -8, 4, 11};
Abs[x]
```

## Output :
{1, 4, 8, 4, 11}

## Programming : 3
```
Absolute1[x_] := Module[{}, If[x < 0, -x, x]]

Absolute1[-5]
```

## Output :
5

## Programming : 4
Y[x_] := $\begin{cases} -x & x < 0 \\ x & x \geq 0 \end{cases}$

```
Y[-5]
```

## Output :
5

# PROGRAM # 21
# "AREAS AND PARAMETERS OF RECTANGLE, CIRCLE, SQUARE"

### Aim :
To find areas & parameters of rectangle ,circle ,square

### Programming : 1
```
Arectangle[l_, b_] := l*b

Arectangle[2, 3]
```

### Output :
6

### Programming : 2
```
Prectangle[l_, b_] := 2 (l + b)

Prectangle[5, 4]
```

### Output :
18

### Programming : 3
```
Acircle[r_] :=π *r^2

Acircle[6]
```

### Output :
36π

### Programming : 4
```
Pcircle[r_] := 2*π*r

Pcircle[4]
```

### Output :
8π

### Programming : 5
```
Asquare[a_] := a^2

Asquare[7]
```

### Output :
49

## Programming : 6

```
Psquare[a_] := 4*a

Psquare[16]
```

## Output :

64

## PROGRAM # 22
## "FACTORIAL OF A NATURAL NUMBER"

### Aim :
To find the factorial of a natural number

### Programming : 1
```
Factorial[5]
```

### Output :
120

### Programming : 2
```
Fact[n_] := Module[{}, p = 1;
  For[i = 1, i <= n, i = i + 1, p = p*i]; Print[p];]
```

(i) Fact[5]

### Output :
120

(ii) Fact[12]

### Output :
479001600

(iii) Fact[7]

### Output :
5040

# PROGRAM # 23
# "WHETHER INTEGER IS EVEN OR ODD"

## Aim :
To read an integer and check whether it is even or odd

## Programming:
```
Evenodd[a_] := Module[{},
  If[Mod[a, 2] == 0, Print["Number is even"],
    Print["Number is odd"];];]
```

**(i)** Evenodd[40]

## Output :
Number is even

**(ii)** Evenodd[35]

## Output :
Number is odd

**(iii)** Evenodd[7892]

## Output :
Number is even

**(iv)** Evenodd[23]

## Output :
Number is odd

## PROGRAM # 24
## "MAXIMUM ELEMENT IN A LIST"

### Aim :
To find the maximum element in a list

### Programming : 1
```
Max[2, 5, 8, 9, 32]
```

### Output :
32

### Programming : 2
```
Max1[a_] := Module[{}, m1 = a[[1]];
  n = Length[a];
  For[i = 2, i <= n, i = i + 1,
   If[m1 < a[[i]], m1 = a[[i]], Continue;];];
  Print["Maximum element in the list is=", m1];];
```

(i) a = {3, 7, 6, 9};

Max1[a]

### Output :
Maximum element in the list is=9

(ii) a = {34, 67, 91, 20};

Max1[a]

### Output :
Maximum element in the list is=91

(iii) a = {3625, 5487, 9023, 5234};

Max1[a]

### Output :
Maximum element in the list is=9023

## PROGRAM # 25
## "MINIMUM ELEMENT IN A LIST"

### Aim :
To find the minimum element in a list

### Programming : 1
```
Min[2, 5, 8, 9, 32]
```

### Output :
2

### Programming : 2
```
Min1[a_] := Module[{}, m1 = a[[1]];
  n = Length[a];
  For[i = 2, i <= n, i = i + 1,
   If[m1 > a[[i]], m1 = a[[i]], Continue;];];
  Print["Minimum element in the given list is=", m1];];
```

(i) a = {3, 7, 6, 9};

Min1[a]

### Output :
Minimum element in the given list is=3

(ii) a = {57, 78, 45, 89};

Min1[a]

### Output :
Minimum element in the given list is=45

(iii) a = {4867, 2345, 8967, 9456};

Min1[a]

### Output :
Minimum element in the given list is=2345

# PROGRAM # 26
# "FUNCTION DEFINED BY TWO OR THREE CONDITIONS"

## Aim :
To define a function by two or three conditions

## Programming : 1
F[x_] := $\begin{cases} -x & x < 0 \\ x & x \geq 0 \end{cases}$

(i) F[2]

## Output :
2

(ii) F[-9]

## Output :
9

(iii) F[0]

## Output :
0

## Programming : 2
$f[\text{x\_}] := \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ 0 & x = 0 \end{cases}$

(i) f[-2]

## Output :
-1

(ii) f[2]

## Output :
1

(iii) f[0]

## Output :
0