Date :
Page No. : 1

BCSL-58

Prog 1 :=

```
# include <bits/stdc++.h>
using namespace std;
# define N 3 // Number of unknown
int forwardElim(double mat[N][N+1]);
Void backSub(double mat[N][N+1]);
Void gaussianElimination(double mat[N][N+1])
{
int Singular_flag = forwardElim(mat);
if (Singular matrix.\n");
if (mat[singular flag][N])
printf("Inconsistent System.");
  else
printf("May have infinitely many" "Solution");
  return;
}
backSub(mat);
}
Void Swap_row(double mat[N][N+1], int i, int j)
{
for (int k=0; k<=N; k++)
{
double temp = mat[i][k];
mat[i][k] = mat[j][k];
  mat[j][k] = temp;
}
```

```
}
Void Print (double mat[N][N+1])
 {
 for (int i=0; i<N; i++, Printf("\n"))
    for (int j=0; j<=N; j++)
      Printf("%6f ", mat[i][j]);
    Printf("\n");
 }
int forward elim(double mat[N][N+1])
 {
 for (int k=0; k<N; k++)
{
 int i_max = k;
 int v_max = mat[i_max][k];
  for (int i = k+1; i<N; i++)
  if (abs(mat[i][k]) > v_max)
   v_max = mat[i][k], i_max = i;
if (!max[k][i_max])
 return k; //Matrix is Singular
  if(i_max != k)
 Swap_row (mat, k, i_max);
 for (int i = k+1; i<N; i++)
   {
   double f = mat[i][k]/mat[k][k];
   for (int j = k+1; j<=N; j++)
   mat Ei][j] = mat[k][j]*f;
/* filling lower triangular matrix with zeros*/
```

```
mat[i][k] = 0;
 }
 // Print(mat);  //for matrix state
 }
 // Print(mat);   //For matrix state
   return 1;
 }
 Void backSub(double mat[N][N+1])
 {
 double x[N]; //An array to Store Solution
   for (int i = N-1; i>=0; i--)
   {
   /* Start with the RHS of the equation */
     x[i] = mat[i][N];
   for (int j = i+1; j<N; j++)
    {
     x[i] = mat[i][j] * x[j];
    }
    x[i] = x[i]/mat[i][i];
    }
 Printf("\nSolution for the System:\n");
   for (int i = 0; i<N; i++)
      Printf("%0.1f\n", x[i]);
      }
   int main()
   /* input martrix */
```

```
double mat[N][N+1] = {{5.0, -7.0, 8.0, 11.0},
                      {4.0, -9.0, 3.0, 2.0},
                      {9.0, 2, 5.0, 25.0}
                      };
gaussian Elimination (mat);
  return 0;
}
```

Ans2=

```
#include <stdio.h>
#include <math.h>
#define x 2
main()
{
float x[x][x+1], a[x], ae, max, s, se;
int i, j, r, maxit;
for (i=0; i<x; i++) a[i]=0;
puts ("Enter the elements of augmented matrix
  row wise \n");
for (i=0; i<x; i++)
{
for (j=0; j<x+1; j++)
{
Scanf ("%f", &x[i][j]);
  }
  }
printf("Enter the allowed error and maximum number
  of iteration: ");
```

```
Scanf("%f%d", &ae, &mxit);
  printf("\nIteration\tx[1]\tx[2]\n");
    for(r=1; r<=mxit; r++)
    {
    max=0;
  for (i=0; i<x; i++)
    {
    s=0;
  for (j=0; j<x; j++)
  if (j!=i) s+=x[i][j]*a[j];
  t=(x[i][x]-s)/x[i][i];
    e=fabs(a[i]-t);
    a[i]=t;
    }
 Printf("%5d\t", r);
  for(i=0; i<x; i++)
 Printf("%9.4f\t", a[i]);
   Printf("\n");
 if (max<ae)
 {
 printf("Converges in %3d iteration\n", r);
    for(i=0; i<x; i++)
 Printf("a[%3d]=%7.4f\n", i+1, a[i]);
   return 0;
   }
 }
}
```

Date :
Page No. : 6

Ans 3 =

```
// C++ Program for implementation of Newton Raphson Method for
// Solving equations
#include <bits/stdc++.h>
#define EPSILON 0.001
using namespace std;
// An example function whose solution is determined using
// Bisection method. The function is 2x^4 - 5x^2 + 10x + 32
double func(double x)
{
return 2*x*x*x*x - 5*x*x + 10*x;
}
// Derivative of the above function which is 8x^3 - 10x + 10
double deriv func(double x)
{
return 8*x*x*x - 10*x + 10;
}
// Function to find the root
void newton Raphson(double x)
{
double h = func(X) / deriv func(X);
while (abs(h) >= EPSILON)
{
```

```
{
h = func(x)/derivfunc(x);
// x(i+1) = x(i) - f(x)/f'(x)
x = x - h;
}
Cout << " The value of the root is: " << x;
}
// Driver Program to test above
int main()
{
double x0 = -20; // initial values assumed
netton Raphson(x0);
return 0;
}
```

Ans 4:

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
int main()
{
float x[10], y[10], temp = 1, f[10], sum, f;
int i, n, j, k = 0, c;
printf("\n how many record u/or will be enter: ");
scanf("%d", &n);
for (i = 0; i < n; i++)
{
```

Date :
Page No. : 8

```
printf("\n\nenter the value of x%d: ",i);
scanf("%f", &x[i]);
printf("\n\nenter the value of f(x%d): ",i);
scanf("%f", &y[i]);
}
printf("\n\n\n Enter x for finding f(x): ");
scanf("%f", &p);
for(i=0; i<n; i++)
{
temp = i;
k = i;
for(j=0; j<n; j++)
{
if(k==j)
{
continue;
}
else
{
temp = temp * ((p-x[j])/(x[k]-x[j]));
}
}
f[i] = y[i] * temp;
}
for(i=0; i<n; i++)
{
Sum = Sum + f[i];
```

```
}
printf("\n\n Value of f(%.1f) = %f", p, sum);
}
```

Ans 5:-

```
// C++ Program to interpolate using
// newton's forward interpolation
#include <bits/stdc++.h>
using namespace std;
// Calculating u mentioned in the formula
float u_cal(float u, int n)
{
    float temp = u;
    for (int i = 1; i < n; i++)
        temp = temp * (u - i);
    return temp;
}
// Calculating factorial of given number n
int fact(int n)
{
    int f = 1;
    for (int i = 2; i <= n; i++)
        f *= i;
    return f;
}
int main()
{
// Number of values given
```

Date :
Page No. : 10

```
int n = 4;
float x[] = {45, 50, 55, 60};
  // y[][] is used for difference table
  // with y[][0] used for input
    float y[n][n];
  y[0][0] = 0.7071;
  y[1][0] = 0.7660;
  y[2][0] = 0.8192;
  y[3][0] = 0.8660;
  // Calculating the forward difference
    // table
  for (int i = 1; i < n; i++) {
    for (int j = 0; j < n - i; j++)
      y[j][i] = y[j+1][i-1] - y[j][i-1];
  }
  // Displaying the forward difference table
  for (int i = 0; i < n; i++) {
    Cout << setw (4) << x[i]
      << "\t";
  for (int j = 0; j < n - i; j++)
    Cout << setw(4) << y[i][j]
      << "\t";
    Cout << endl;
  }
  // Value to Interpolate at
    float value = 52;
  // Initializing u and Sum
```

```
float Sum = y[0][0];
float u = (value - x[0]) / (x[1]-x[0]);
for (int i = 1; i < n; i++){
  Sum = Sum + (u - cal(u,i)* y[0][i])/
        fact (i);
}
Cout <<"\n value at " << value << "is"
     << sum << endl;
  return 0;
}
```

Ans 6 :

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
int main()
{
float x[10], y[10][10], Sum, f, u, temp;
 int i, n, j, k = 0, f, m;
 float fact (int);
Printf ("\n how many record you will be enter:");
Scanf ("%d", &n);
 for (i= 0; i<n; i++)
  {
  Printf ("\n\n enter the value of x%d: ", i);
Scanf ("%f", &x[i]);
Printf ("\n\n enter the value of f(x%d): ", i);
```

```
Scanf("%f", &y[k][i]);
 }
printf("\n\nEnter x for finding f(x): ");
   Scanf("%f", &f);
   for (i=1; i<n; i++)
     {
      for (j=0; j<n-i; j++)
      {
     y[i][j] = y[i-1][j+1] - y[i-1][j];
      }
 }
printf("\n ____________ \n");
printf("\n x(p) \t y(i) \t y1(i) y2(i) y3(i) y4(i)");
printf("\n ____________ \n");
  for (i = 0; i <n; i++)
    {
    printf("\n %.3f", x[i]);
    for (j= 0; j <n-i; j++)
  {
   printf(" ");
printf("%.3f", y[j][i]);
  }
printf("\n");
 }
i = 0;
 do
{
```

```
if (x[i] <= p && p < x[i+1])
    k=1
    else
    i++;
}
while (k != 1);
  f=i;
 u=(p-x[f])/(x[f+1]-x[f]);
  printf ("\n\n u = %.3f", u);
 n= n-i+1;
 Sum = 0;
for (i=0; i<n-1; i++)
   {
   temp=1;
for (j=0; j<i; j++)
{
 temp = temp * (u-j);
  }
n=fact(i);
Sum =Sum + temp * (y[i][f]/n);
 }
printf("\n\n f(%.2f) = %f", p, Sum);
 }
float fact (int a)
   {
   float fac=1;
 if (a==0)
```

```
return (1);
 else
fac = a* fact (a-1);
  return (fac);
 }
```

Ans 7:

```
# include <stdio.h>
# include <math.h>
/* Define the function to be integrated here '2x^2+x+5'*/
double f(double x){
 return 2* x*x+x+5;
 }
/* Program begins */
  main(){
 int n, i;
double a, b, h, x, sum=0, integral;
/* Ask the user for necessary input */
printf ("\nEnter the no. of sub-intervals (EVEN):");
 Scanf("%d", &n);
 printf("\nEnter the initial limit : ");
 Scanf("%lf", &a);
 Printf ("\nEnter the final limit :");
 Scanf ("%lf", &b);
/* Begin Simpsons procedure :*/
   h = fabs (b- a)/n;
 for(i= 1; i<n; i++){
x = a+i*h;
```

```
if (i %2 == 0){
  Sum = Sum + 2 * f(x);
  }
  else{
  Sum = Sum + 4 * f(x);
  }
  }
  intergual = (h/3) * (f(a) + f(b) + Sum);
    /* printf the answer */
Printf ("\n the intergual is : %.4f \n", intergual);
  }
```