Graphic Era
Deemed to be University

Term work

on

Computer Based Numerical and Statistical Techniques

(PMA 402) 2020-21

Submitted to:

Mr. Piyush Aggarwal

Assistant Professor

GEU D.Dun

Submitted by:

SHOURYA KAPOOR

Roll.No.:1918914

Class Roll.No./Section: 84/IT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA UNIVERSITY, DEHRADUN

ACKNOWLEDGMENT

I would like to particularly thank my CBNST Lab Faculty **Mr Piyush Aggarwal** for his patience, support and encouragement throughout the completion of this Term work.

At last but not the least I greatly indebted to all other persons who directly or indirectly helped me during this course.

SHOURYA

Uni. Roll No.- 1918914

B.Tech IT-IV Sem

Session: 2020-2021

GEU Dehradun

Table of Contents

Program 1

Objective: Program to Implement Bisection Method
Algorithm:

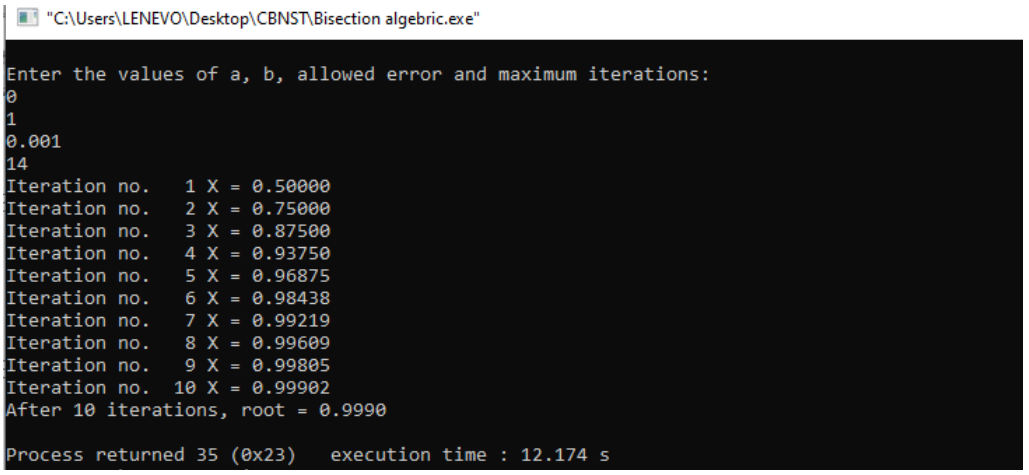
1. Start
2. Read x1, x2, e
   *Here x1 and x2 are initial guesses
   e is the absolute error i.e. the desired degree of accuracy*
3. Compute: f1 = f(x1) and f2 = f(x2)
4. If (f1*f2) > 0, then display initial guesses are wrong and goto (11).
   Otherwise continue.
5. x = (x1 + x2)/2
6. If ([(x1 – x2)/x] < e), then display x and goto (11).
   * Here [ ] refers to the modulus sign. *
7. Else, f = f(x)
8. If ((f*f1) > 0), then x1 = x and f1 = f.
9. Else, x2 = x and f2 = f.
10. Goto (5).
    Now the loop continues with new values.
11. Stop

Source Code

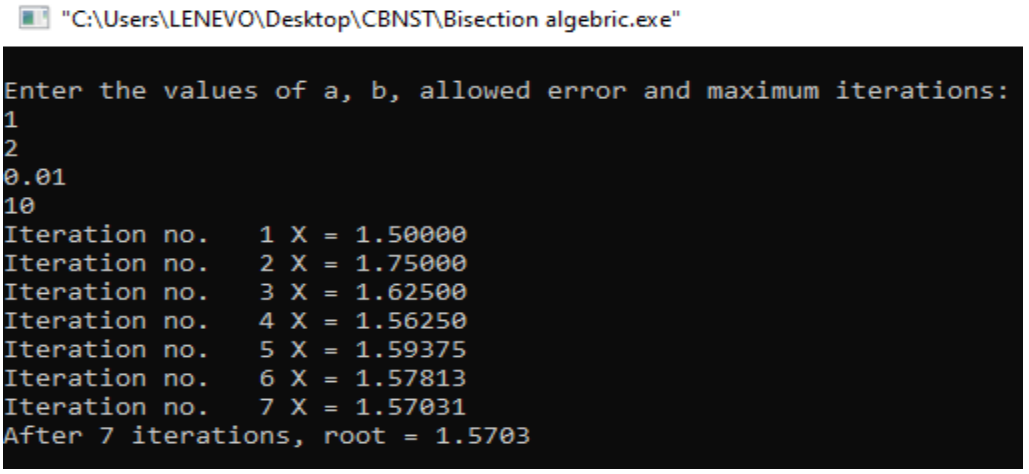
```
#include<stdio.h>
#include<math.h>
float fun (float x)
{
   return (x*x*x - 4*x - 9);
            cos(x) - x * exp(x)
            x=cos(x)
}
void bisection (float *x, float a, float b, int *itr)
/* this function performs and prints the result of one iteration */
{
   *x=(a+b)/2;
   ++(*itr);
   printf("Iteration no. %3d X = %7.5f\n", *itr, *x);
}
void main ()
{
   int itr = 0, maxmitr;
   float x, a, b, allerr, x1;
   printf("\nEnter the values of a, b, allowed error and maximum
iterations:\n");
   scanf("%f %f %f %d", &a, &b, &allerr, &maxmitr);
   bisection (&x, a, b, &itr);
   do
   {
      if (fun(a)*fun(x) < 0)
         b=x;
      else
         a=x;
      bisection (&x1, a, b, &itr);
      if (fabs(x1-x) < allerr)
```

```
        {
            printf("After %d iterations, root = %6.4f\n", itr, x1);
            return 0;
        }
        x=x1;
    }
    while (itr < maxmitr);
    printf("The solution does not converge or iterations are not
sufficient");
    return 1;
}
```

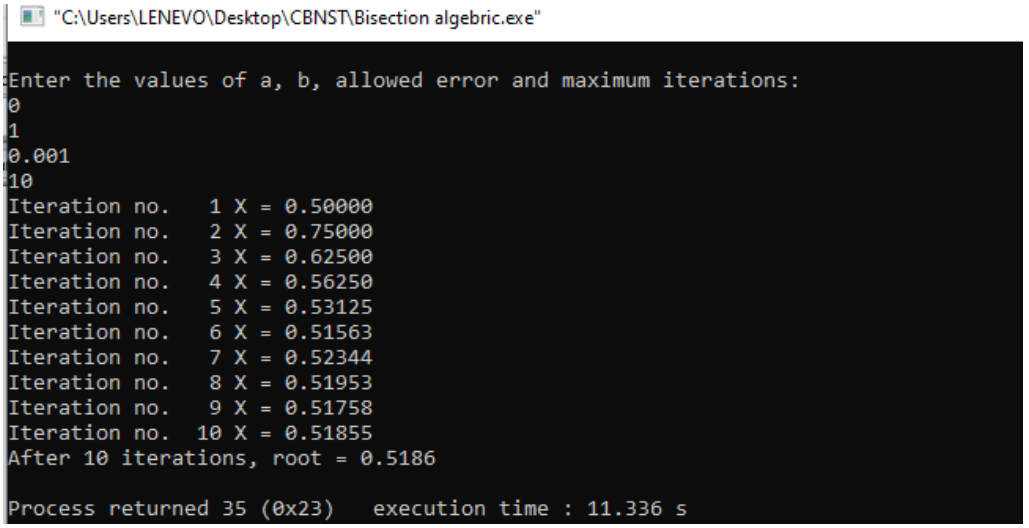
Output



```
"C:\Users\LENEVO\Desktop\CBNST\Bisection algebric.exe"

Enter the values of a, b, allowed error and maximum iterations:
0
1
0.001
14
Iteration no.   1 X = 0.50000
Iteration no.   2 X = 0.75000
Iteration no.   3 X = 0.87500
Iteration no.   4 X = 0.93750
Iteration no.   5 X = 0.96875
Iteration no.   6 X = 0.98438
Iteration no.   7 X = 0.99219
Iteration no.   8 X = 0.99609
Iteration no.   9 X = 0.99805
Iteration no.  10 X = 0.99902
After 10 iterations, root = 0.9990

Process returned 35 (0x23)   execution time : 12.174 s
```



```
"C:\Users\LENEVO\Desktop\CBNST\Bisection algebric.exe"

Enter the values of a, b, allowed error and maximum iterations:
1
2
0.01
10
Iteration no.    1 X = 1.50000
Iteration no.    2 X = 1.75000
Iteration no.    3 X = 1.62500
Iteration no.    4 X = 1.56250
Iteration no.    5 X = 1.59375
Iteration no.    6 X = 1.57813
Iteration no.    7 X = 1.57031
After 7 iterations, root = 1.5703
```



```
"C:\Users\LENEVO\Desktop\CBNST\Bisection algebric.exe"

Enter the values of a, b, allowed error and maximum iterations:
0
1
0.001
10
Iteration no.    1 X = 0.50000
Iteration no.    2 X = 0.75000
Iteration no.    3 X = 0.62500
Iteration no.    4 X = 0.56250
Iteration no.    5 X = 0.53125
Iteration no.    6 X = 0.51563
Iteration no.    7 X = 0.52344
Iteration no.    8 X = 0.51953
Iteration no.    9 X = 0.51758
Iteration no.   10 X = 0.51855
After 10 iterations, root = 0.5186

Process returned 35 (0x23)   execution time : 11.336 s
```

Program 2

Objective: Program to Implement Regular Falsi Method
Algorithm:

1. Start
2. Read values of x0, x1 and e
   *Here x0 and x1 are the two initial guesses
   e is the degree of accuracy or the absolute error i.e. the stopping criteria*
3. Computer function values f(x0) and f(x1)
4. Check whether the product of f(x0) and f(x1) is negative or not.
   If it is positive take another initial guesses.
   If it is negative then goto step 5.
5. Determine:
   x = [x0*f(x1) – x1*f(x0)] / (f(x1) – f(x0))
6. Check whether the product of f(x1) and f(x) is negative or not.
   If it is negative, then assign x0 = x;
   If it is positive, assign x1 = x;
7. Check whether the value of f(x) is greater than 0.00001 or not.
   If yes, goto step 5.
   If no, goto step 8.
   Here the value 0.00001 is the desired degree of accuracy, and hence the stopping criteria.
8. Display the root as x.
9. Stop

Source Code

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   return cos(x) - x*exp(x);
          (x*x*x - 4*x - 9)
          x=cos(x)
}
void regula (float *x, float x0, float x1, float fx0, float fx1, int *itr)
{
   *x = x0 - ((x1 - x0) / (fx1 - fx0))*fx0;
   ++(*itr);
   printf("Iteration no. %3d X = %7.5f \n", *itr, *x);
}
void main ()
{
   int itr = 0, maxmitr;
   float x0,x1,x2,x3,allerr;
   printf("\nEnter the values of x0, x1, allowed error and maximum iterations:
\n");
   scanf("%f %f %f %d", &x0, &x1, &allerr, &maxmitr);
   regula (&x2, x0, x1, f(x0), f(x1), &itr);
   do
   {
      if (f(x0)*f(x2) < 0)
         x1=x2;
      else
         x0=x2;
      regula (&x3, x0, x1, f(x0), f(x1), &itr);
      if (fabs(x3-x2) < allerr)
      {
         printf("After %d iterations, root = %6.4f\n", itr, x3);
         return 0;
      }
      x2=x3;
   }
   while (itr<maxmitr);
   printf("Solution does not converge or iterations not sufficient:\n");
   return 1;
```

}

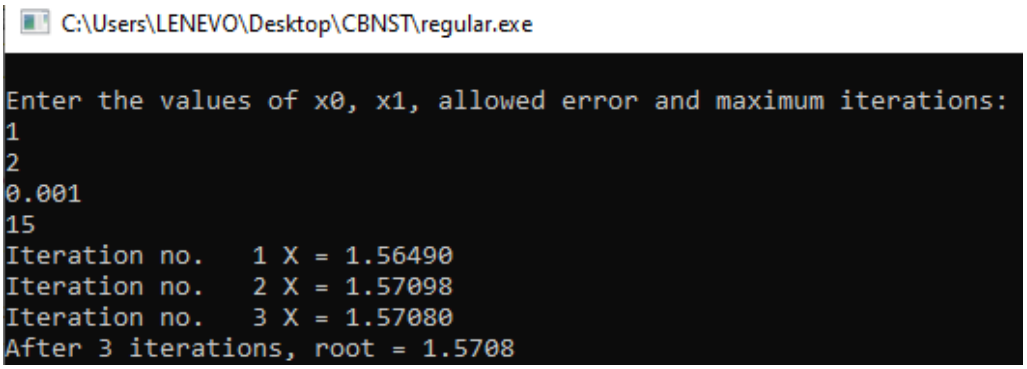
Output

C:\Users\LENEVO\Desktop\CBNST\regular.exe

```
Enter the values of x0, x1, allowed error and maximum iterations:
0
1
0.001
9
Iteration no.   1 X = 0.31467
Iteration no.   2 X = 0.44673
Iteration no.   3 X = 0.49402
Iteration no.   4 X = 0.50995
Iteration no.   5 X = 0.51520
Iteration no.   6 X = 0.51692
Iteration no.   7 X = 0.51748
After 7 iterations, root = 0.5175

Process returned 34 (0x22)   execution time : 9.333 s
```

C:\Users\LENEVO\Desktop\CBNST\regular.exe

```
Enter the values of x0, x1, allowed error and maximum iterations:
1
2
0.01
15
Iteration no.   1 X = 5.00000
Iteration no.   2 X = 1.44444
Iteration no.   3 X = 1.83259
Iteration no.   4 X = 2.13615
Iteration no.   5 X = 2.35128
Iteration no.   6 X = 2.49248
Iteration no.   7 X = 2.58033
Iteration no.   8 X = 2.63312
Iteration no.   9 X = 2.66417
Iteration no.  10 X = 2.68221
Iteration no.  11 X = 2.69260
Iteration no.  12 X = 2.69856
After 12 iterations, root = 2.6986

Process returned 35 (0x23)   execution time : 4.822 s
Press any key to continue.
```



Program 3

Objective: Program to Implement Secant Method

Algorithm:

1. Start
2. Get values of x0, x1 and e
   *Here x0 and x1 are the two initial guesses
   e is the stopping criteria, absolute error or the desired degree of accuracy*
3. Compute f(x0) and f(x1)
4. Compute x2 = [x0*f(x1) – x1*f(x0)] / [f(x1) – f(x0)]
5. Test for accuracy of x2
   If [(x2 – x1)/x2] > e, *Here [ ] is used as modulus sign*
   then assign x0 = x1 and x1 = x2
   goto step 4
   Else,
   goto step 6
6. Display the required root as x2.
7. Stop

Source Code

```
#include<stdio.h>
float f(float x)
{
   return cos(x) - x*exp(x);
          (x*x*x - 4*x - 9)
          x=cos(x)
 }
float main()
{
   float a,b,c,d,e;
   int count=1,n;
   printf("\n\nEnter the values of a and b:\n"); //(a,b) must contain the solution.
   scanf("%f%f",&a,&b);
   printf("Enter the values of allowed error and maximun number of iterations:
\n");
   scanf("%f %d",&e,&n);
   do
   {
      if(f(a)==f(b))
      {
         printf("\nSolution cannot be found as the values of a and b are same.
\n");
      return;
      }
      c=(a*f(b)-b*f(a))/(f(b)-f(a));
      a=b;
      b=c;
      printf("Iteration No-%d    x=%f\n",count,c);
      count++;
      if(count==n)
      {
```

```
        break;
        }
    } while(fabs(f(c))>e);
    printf("\n The required solution is %f\n",c);

}
```

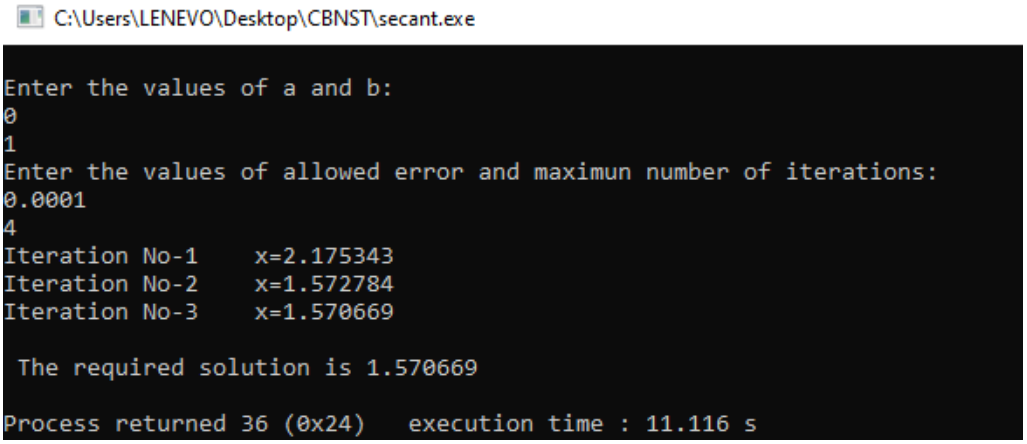
Output

C:\Users\LENEVO\Desktop\CBNST\secant.exe

```
Enter the values of a and b:
1
2
Enter the values of allowed error and maximun number of iterations:
0.01
10
Iteration No-1    x=0.832673
Iteration No-2    x=0.728779
Iteration No-3    x=0.562401
Iteration No-4    x=0.524782
Iteration No-5    x=0.518014

 The required solution is 0.518014

Process returned 36 (0x24)   execution time : 10.369 s
```

C:\Users\LENEVO\Desktop\CBNST\secant.exe

```
Enter the values of a and b:
1
2
Enter the values of allowed error and maximun number of iterations:
0.001
5
Iteration No-1    x=5.000000
Iteration No-2    x=2.257143
Iteration No-3    x=2.431810
Iteration No-4    x=2.779584

 The required solution is 2.779584

Process returned 36 (0x24)   execution time : 7.428 s
```



Program 4

Objective: Program to Implement Iteration Method

Algorithm:

1. Start
2. Read values of x0 and e.
   *Here x0 is the initial approximation
   e is the absolute error or the desired degree of accuracy, also the stopping criteria*
3. Calculate x1 = g(x0)
4. If [x1 – x0] <= e, goto step 6.
   Here [ ] refers to the modulus sign
5. Else, assign x0 = x1 and goto step 3.
6. Display x1 as the root.
7. Stop

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define   f(x)   cos(x)-3*x+1
#define   g(x)  (1+cos(x))/3
                 x*x*x – x -1

int main()
{
        int step=1, N;
        float x0, x1, e;
        printf("Enter initial guess: ");
        scanf("%f", &x0);
        printf("Enter tolerable error: ");
        scanf("%f", &e);
        printf("Enter maximum iteration: ");
        scanf("%d", &N);
        printf("\nStep\tx0\t\tf(x0)\t\tx1\t\tf(x1)\n");
        do
        {
                x1 = g(x0);
                printf("%d\t%f\t%f\t%f\t%f\n",step, x0, f(x0), x1, f(x1));

                step = step + 1;

                if(step>N)
                {
                        printf("Not Convergent.");
                        exit(0);
                }

                x0 = x1;

        }while( fabs(f(x1)) > e);

        printf("\nRoot is %f", x1);
```

```
        getch();
        return(0);
}
```

Output

```
C:\Users\LENEVO\Desktop\CBNST\Iteration.exe
Enter initial guess: 1
Enter tolerable error: 0.01
Enter maximum iteration: 5

Step    x0              f(x0)           x1              f(x1)
1       1.000000        -1.459698       0.513434        0.330761
2       0.513434        0.330761        0.623688        -0.059333
3       0.623688        -0.059333       0.603910        0.011391
4       0.603910        0.011391        0.607707        -0.002162

Root is 0.607707
```

```
C:\Users\LENEVO\Desktop\CBNST\Iteration.exe
Enter initial guess: 1
Enter tolerable error: 0.01
Enter maximum iteration: 6

Step    x0              f(x0)           x1              f(x1)
1       1.000000        -1.000000       1.000000        -1.000000
2       1.000000        -1.000000       1.000000        -1.000000
3       1.000000        -1.000000       1.000000        -1.000000
4       1.000000        -1.000000       1.000000        -1.000000
5       1.000000        -1.000000       1.000000        -1.000000
6       1.000000        -1.000000       1.000000        -1.000000
Not Convergent.
Process returned 0 (0x0)   execution time : 9.320 s
```

Program 5

Objective: Program to finding errors.

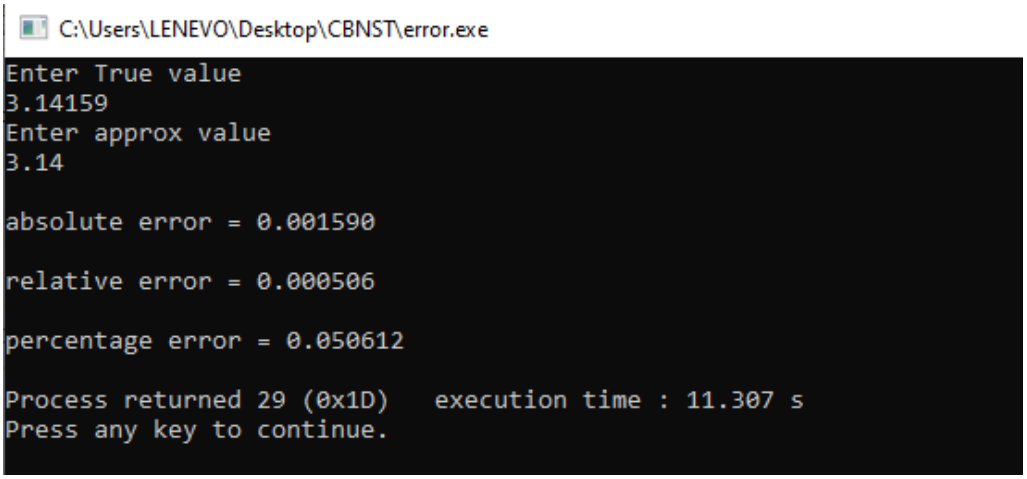
Algorithm:

1. Start
2. Take input say True value and Approximate value
3. Calculate absolute error using formula-
    a. Absolute error= True value -Approximate value
4. Calculate relative error using formula-
    a. relative error= Approximate value/ True value
5. Calculate percentage error using formula
    a. Percentage error=relative error *100
6. Stop

Source Code

```
#include<stdio.h>
#include<math.h>
void main()
{
   float abserror, relerror, percerror, trueval, approxval;
   printf("Enter True value");
   scanf("%f",&trueval);
   printf("Enter approx value");
   scanf("%f",&approxval);
   abserror=fabs(trueval-approxval);
   relerror=abserror/trueval;
   percerror=relerror*100;
   printf("\nabsolute error = %f\n",abserror);
   printf("\nrelative error = %f\n",relerror);
   printf("\npercentage error = %f\n",percerror);
}
```

Output


C:\Users\LENEVO\Desktop\CBNST\error.exe
Enter True value
3.14159
Enter approx value
3.14
absolute error = 0.001590
relative error = 0.000506
percentage error = 0.050612
Process returned 29 (0x1D)   execution time : 11.307 s
Press any key to continue.

Program 6

Objective: Program to implement Newton Raphson Method.

Algorithm:

1. Start
2. Read x, e, n, d
   *x is the initial guess
   e is the absolute error i.e the desired degree of accuracy
   n is for operating loop
   d is for checking slope*
3. Do for i =1 to n in step of 2
4. f = f(x)
5. f1 = f'(x)
6. If ([f1] < d), then display too small slope and goto 11.
   [ ] is used as modulus sign
7. x1 = x – f/f1
8. If ([(x1 – x)/x1] < e), the display the root as x1 and goto 11.
   [ ] is used as modulus sign
9. x = x1 and end loop
10. Display method does not converge due to oscillation.
11. Stop

Source Code

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   return x*log10(x) - 1.2;
            x * x * x - x * x + 2;
           3*x - cos(x) -1 ;
}
float df (float x)
{
   return log10(x) + 0.43429;
           3 * x * x - 2 * x;
           3 + sin(x);
}
void main()
{
   int itr, maxmitr;
   float h, x0, x1, allerr;
   printf("\nEnter x0, allowed error and maximum iterations\n");
   scanf("%f %f %d", &x0, &allerr, &maxmitr);
   for (itr=1; itr<=maxmitr; itr++)
   {
      h=f(x0)/df(x0);
      x1=x0-h;
      printf(" At Iteration no. %3d, x = %9.6f\n", itr, x1);
      if (fabs(h) < allerr)
      {
```

```
            printf("After %3d iterations, root = %8.6f\n", itr, x1);
            return 0;
        }
        x0=x1;
    }
      printf(" The required solution does not converge or iterations
are insufficient\n");
    return 1;
}
```

Output

"C:\Users\LENEVO\Desktop\CBNST\newton raphson.exe"

```
Enter x0, allowed error and maximum iterations
2
0.001
5
 At Iteration no.   1, x =  2.813170
 At Iteration no.   2, x =  2.741109
 At Iteration no.   3, x =  2.740646
After   3 iterations, root = 2.740646

Process returned 38 (0x26)   execution time : 8.218 s
```

"C:\Users\LENEVO\Desktop\CBNST\newton raphson.exe"

```
Enter x0, allowed error and maximum iterations
1
0.01
6
 At Iteration no.   1, x = -1.000000
 At Iteration no.   2, x = -1.000000
After   2 iterations, root = -1.000000

Process returned 39 (0x27)   execution time : 10.037 s
```

"C:\Users\LENEVO\Desktop\CBNST\newton raphson.exe"

```
Enter x0, allowed error and maximum iterations
1
0.0001
5
 At Iteration no.   1, x =  0.620016
 At Iteration no.   2, x =  0.607121
 At Iteration no.   3, x =  0.607102
After   3 iterations, root = 0.607102

Process returned 38 (0x26)   execution time : 9.051 s
```

Program 7

Objective: Program to implement Gauss Elimination Method.

Algorithm:

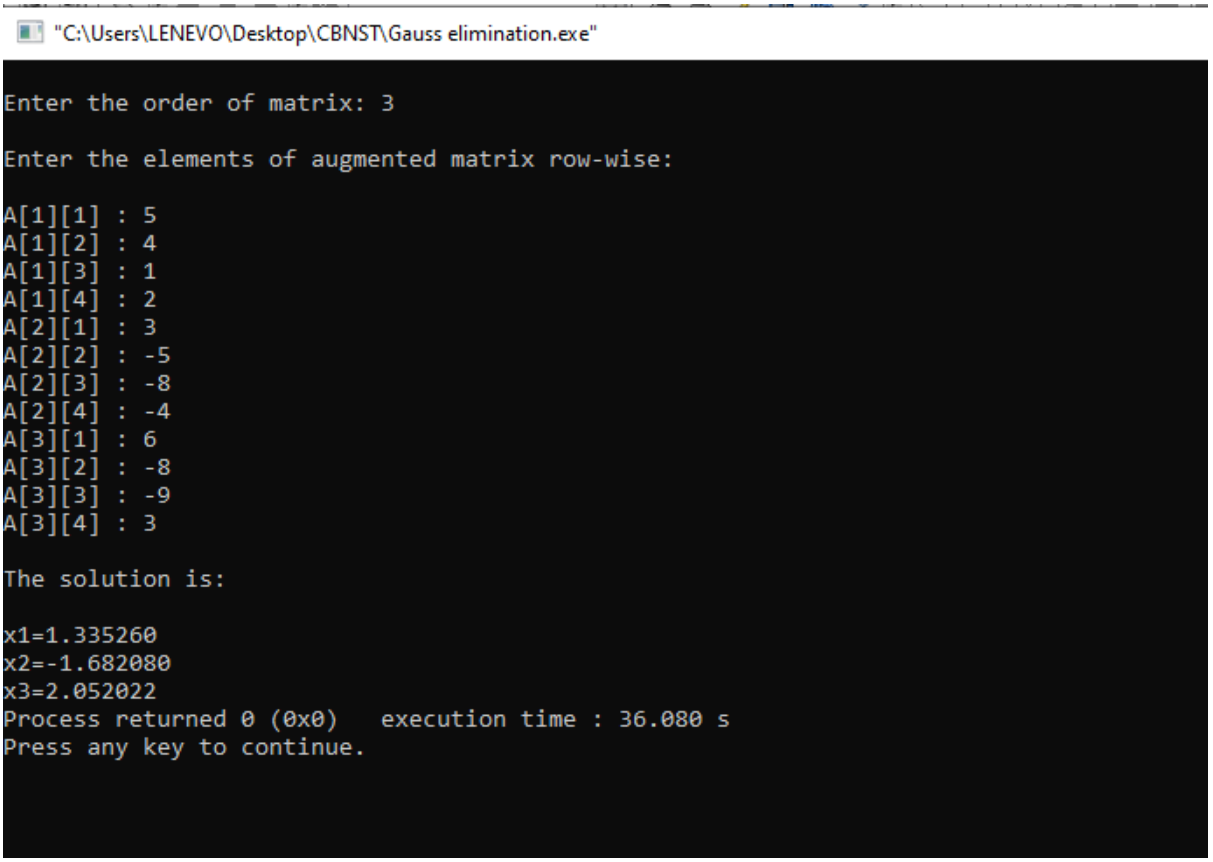
1. Start
2. Declare the variables and read the order of the matrix n.
3. Take the coefficients of the linear equation as:
   Do for k=1 to n
   Do for j=1 to n+1
   Read a[k][j]
   End for j
   End for k
4. Do for k=1 to n-1
   Do for i=k+1 to n
   Do for j=k+1 to n+1
   a[i][j] = a[i][j] – a[i][k] /a[k][k] * a[k][j]
   End for j
   End for i
   End for k
5. Compute x[n] = a[n][n+1]/a[n][n]
6. Do for k=n-1 to 1
   sum = 0
   Do for j=k+1 to n
   sum = sum + a[k][j] * x[j]
   End for j
   x[k] = 1/a[k][k] * (a[k][n+1] – sum)
   End for k
7. Display the result x[k]
8. Stop

Source Code

```
#include<stdio.h>
int main()
{
   int i,j,k,n;
   float A[20][20],c,x[10],sum=0.0;
   printf("\nEnter the order of matrix: ");
   scanf("%d",&n);
      printf("\nEnter the elements of augmented matrix row-wise: \n\n");
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=(n+1); j++)
      {
         printf("A[%d][%d] : ", i,j);
         scanf("%f",&A[i][j]);
      }
   }
    for(j=1; j<=n; j++) /* loop for the generation of upper triangular matrix*/
   {
      for(i=1; i<=n; i++)
      {
         if(i>j)
         {
            c=A[i][j]/A[j][j];
            for(k=1; k<=n+1; k++)
            {
               A[i][k]=A[i][k]-c*A[j][k];
            }
```

```
            }
        }
    }
    x[n]=A[n][n+1]/A[n][n];
    for(i=n-1; i>=1; i--)
    {
        sum=0;
        for(j=i+1; j<=n; j++)
        {
            sum=sum+A[i][j]*x[j];
        }
        x[i]=(A[i][n+1]-sum)/A[i][i];
    }
    printf("\nThe solution is: \n");
    for(i=1; i<=n; i++)
    {
        printf("\nx%d=%f\t",i,x[i]);
}
    return(0);
}
```

Output



Program 8

Objective: Program to implement Gauss Jordan Method.

Algorithm:

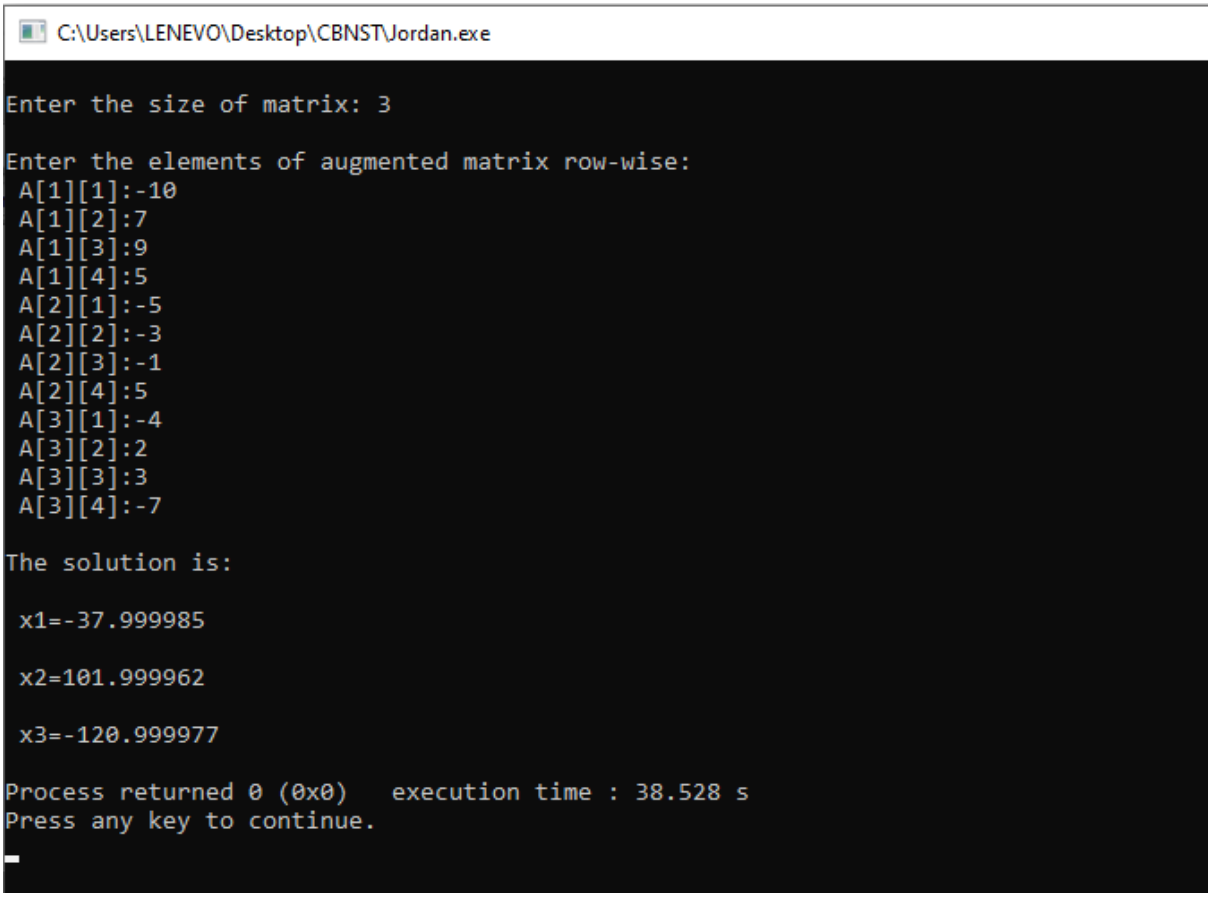
1. Start
2. Read the order of the matrix 'n' and read the coefficients of the linear equations.
3. Do for k=1 to n
   Do for l=k+1 to n+1
   a[k][l] = a[k][l] / a[k][k]
   End for l
   Set a[k][k] = 1
   Do for i=1 to n
   if (i not equal to k) then,
   Do for j=k+1 to n+1
   a[i][j] = a[i][j] – (a[k][j] * a[i][k])
   End for j
   End for i
   End for k
4. Do for m=1 to n
   x[m] = a[m][n+1]
   Display x[m]
   End for m
5. Stop

Source Code

```
#include<stdio.h>
int main()
{
```

```
    int i,j,k,n;
    float A[20][20],c,x[10];
    printf("\nEnter the size of matrix: ");
    scanf("%d",&n);
    printf("\nEnter the elements of augmented matrix row-wise:\n");
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=(n+1); j++)
        {
            printf(" A[%d][%d]:", i,j);
            scanf("%f",&A[i][j]);
        }
    }
    for(j=1; j<=n; j++)
    {
        for(i=1; i<=n; i++)
        {
            if(i!=j)
            {
                c=A[i][j]/A[j][j];
                for(k=1; k<=n+1; k++)
                {
                    A[i][k]=A[i][k]-c*A[j][k];
                }
            }
        }
    }
    printf("\nThe solution is:\n");
    for(i=1; i<=n; i++)
    {
        x[i]=A[i][n+1]/A[i][i];
        printf("\n x%d=%f\n",i,x[i]);
    }
    return(0);
}
```

Output



Program 9

Objective: Program to implement Gauss Jacobi Method.

Algorithm:

1. Start

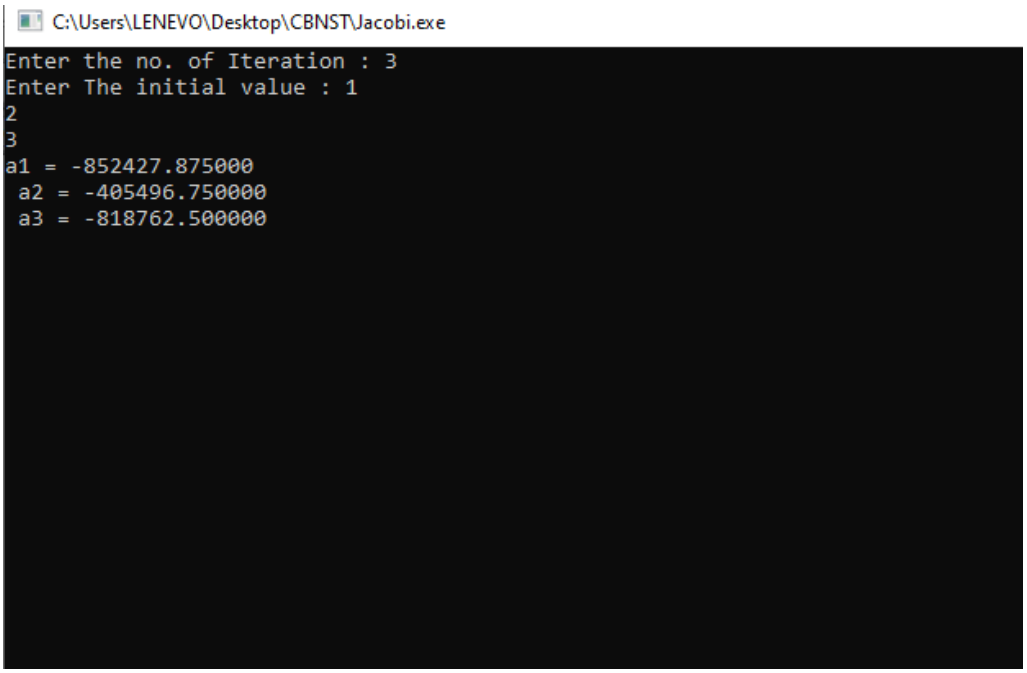
2. Arrange the given system of linear equation in diagonally dominant form
3. Read tolerance error
4. Convert the first equation in terms of first variable,and second equation in term of second variable and so on
5. Set initial guess for x0,y0,z0
6. Substitue value of x0,y0,z0,--------from step 5 in equation obtained on step 4 to calculate new values
7. X1,y1,z1----------
8. If /x0-x1/ > e and /y0-y1/ >e and so on ,then goto step 9
9. Set x0=x1,y0=y1,z0=z1, and so on goto step 6
10. Print value of x1,y1,z1 and so on
11. stop

Source Code

```
#include<stdio.h>
  #include<conio.h>
  #include<math.h>

  float fx(float y,float z)
  {
   float x1; x1=4-2*y-3*z;
   return x1;
  }

  float fy(float x,float z)
  {
   float y1;
   y1=(8-5*x-7*z)/6;
   return y1;
  }

  float fz(float x,float y)
  {
   float z1;
   z1=(3-9*x-y)/2;
   return z1;
  }

  void main()
```

```
{
 int i,j,n;
 float a1,b1,c1;
 float a,b,c;
 float ar[3][4],x[3];
 printf("Enter the no. of Iteration : ");
 scanf("%d",&n);
 printf("Enter The initial value : ");
 scanf("%f %f %f",&a,&b,&c);
 for(i=0;i<n;i++)
 {
    for(j=0;j<n;j++)
    {
       a1=fx(b,c);
       b1=fy(a,c);
       c1=fz(a,b);
       a=a1;
       b=b1;
       c=c1;
    }
 }
 printf("a1 = %f\n a2 = %f\n a3 = %f",a1,b1,c1);

 getch();
}
```

Output


C:\Users\LENEVO\Desktop\CBNST\Jacobi.exe
Enter the no. of Iteration : 3
Enter The initial value : 1
2
3
a1 = -852427.875000
a2 = -405496.750000
a3 = -818762.500000

Program 10

Objective: Program to implement Gauss Seidel Method.

Algorithm:

1. Start
2. Declare the variables and read the order of the matrix n
3. Read the stopping criteria er
4. Read the coefficients aim as
   Do for i=1 to n
   Do for j=1 to n
   Read a[i][j]
   Repeat for j
   Repeat for i
5. Read the coefficients b[i] for i=1 to n
6. Initialize x0[i] = 0 for i=1 to n
7. Set key=0
8. For i=1 to n
   Set sum = b[i]
   For j=1 to n
   If (j not equal to i)
   Set sum = sum – a[i][j] * x0[j]
   Repeat j
   x[i] = sum/a[i][i]
   If absolute value of ((x[i] – x0[i]) / x[i]) > er, then
   Set key = 1
   Set x0[i] = x[i]
   Repeat i

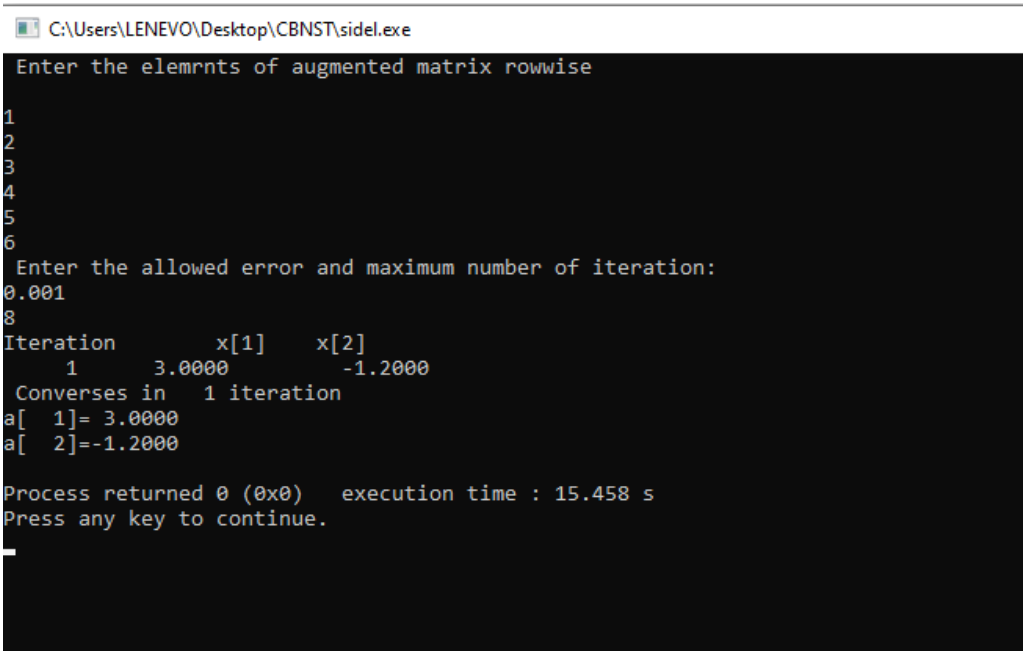
9. If key = 1, then
   Goto step 6
   Otherwise print results

Source Code

```
#include<stdio.h>
#include<math.h>
#define X 2
main()
{
   float x[X][X+1],a[X], ae, max,t,s,e;
   int i,j,r,mxit;
   for(i=0;i<X;i++) a[i]=0;
   puts(" Eneter the elemrnts of augmented matrix rowwise\n");
   for(i=0;i<X;i++)
   {
   for(j=0;j<X+1;j++)
   {
   scanf("%f",&x[i][j]);
   }
   }
   printf(" Eneter the allowed error and maximum number of iteration: ");
   scanf("%f%d",&ae,&mxit);
   printf("Iteration\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(" Converses in %3d iteration\n", r);
        for(i=0;i<X;i++)
        printf("a[%3d]=%7.4f\n", i+1,a[i]);
        return 0;
    }
}}
```

Output



Program 11

Objective: Program to implement Newton Forward Interpolation Method.

Algorithm:

1. Start of the program
2. Input number of terms n
3. Input the array ax
4. Input the array ay
5. h=ax[1] – ax[0]
6. for i=0; i<n-1; i++
7. diff[i] [1]=ay[i + 1] – ay[i]
8. End Loop i
9. for j=2; j<=4; j++
10. for i = 0; i <n – j; i++
11. diff[i][j]=diff [i + 1] [j – 1]-diff [i][j – 1]
12. End Loop i
13. End Loop j
14. i=0
15. Repeat Step 16 until ax[i]<x
16. i=i + 1
17. i=i – 1;

18.p=(x – ax [i])/h

19.y1=p*diff[i – 1][1]

20.y2=p*(p+1)*diff [i – 1][2]/2

21.y3=(p+1)*p*(p-1)*diff[i –2][3]/6

22.y4=(p+2)*(p+1)*p*(p – 1)*diff[i – 3][4]/24

23.y=ay[i]+y1+y2+y3+y4

24.Print output x, y

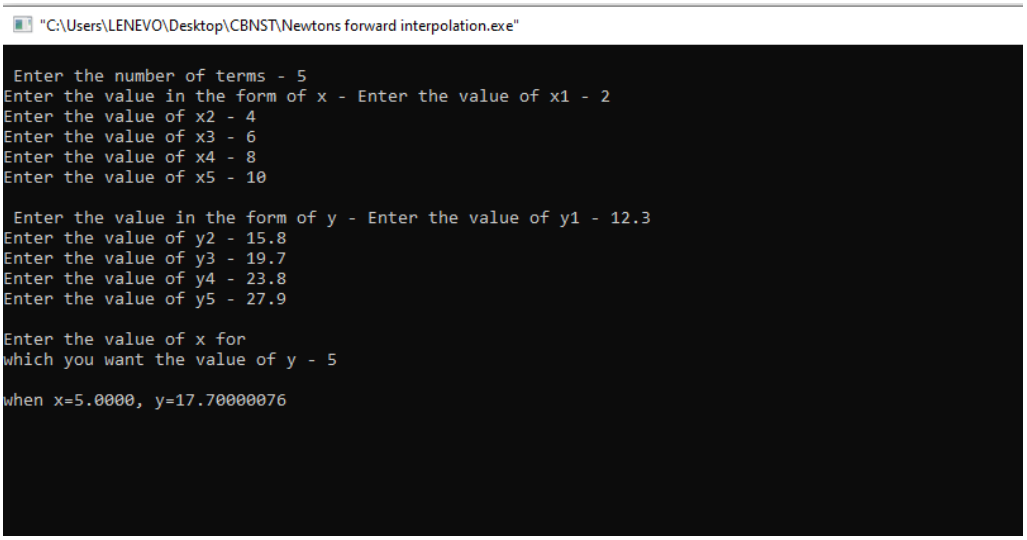
25.End of program.

Source Code

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
void main()
{
    int n;
    int i,j;
    float ax[10];
    float ay[10];
    float x;
    float y = 0;
    float h;
   float p;
   float diff[20][20];
   float y1,y2,y3,y4;
clrscr();
printf("\n Enter the number of terms - ");
scanf("%d",&n);
printf("Enter the value in the form of x - ");
for (i=0;i<n;i++)
{
  printf("Enter the value of x%d - ",i+1);
  scanf("%f",&ax[i]);
}
printf("\n Enter the value in the form of y - ");
for (i=0;i<n;i++)
   {
    printf ("Enter the value of y%d - ", i+1);
    scanf ("%f",&ay [i]);
}
printf("\nEnter the value of x for");
printf("\nwhich you want the value of y - ");
scanf("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
   diff[i][1]=ay[i+1]-ay[i];
      }
```

```
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
  {
     diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
       }
    }
     i=0;
do
    {
     i++;
       }
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i-1][1];
y2=p*(p+1)*diff[i-1][2]/2;
y3=(p+1)*p*(p-1)*diff[i-2][3]/6;
y4=(p+2)*(p+1)*p*(p-1)*diff[i-3][4]/24;
y=ay[i]+y1+y2+y3+y4;

printf("\nwhen x=%6.4f, y=%6.8f ",x,y);
getch();
}
```

Output


"C:\Users\LENEVO\Desktop\CBNST\Newtons forward interpolation.exe"
Enter the number of terms - 5
Enter the value in the form of x - Enter the value of x1 - 2
Enter the value of x2 - 4
Enter the value of x3 - 6
Enter the value of x4 - 8
Enter the value of x5 - 10
Enter the value in the form of y - Enter the value of y1 - 12.3
Enter the value of y2 - 15.8
Enter the value of y3 - 19.7
Enter the value of y4 - 23.8
Enter the value of y5 - 27.9
Enter the value of x for
which you want the value of y - 5
when x=5.0000, y=17.70000076

Program 12

Objective: Program to implement Newton Backward Interpolation Method.

Algorithm:

1. Start of the program.
2. Input number of terms n
3. Input the array ax
4. Input the array ay
5. h=ax[1]-ax[0]
6. for i=0; i<n–1; i++
7. diff[i][1]=ay[i+1]–ay[i]
8. End Loop i
9. for j = 2; j < = 4; j + +
10. for i=0; i<n–j; i++
11. diff[i][j]=diff[i+1][j–1]–diff [i][j–1]
12. End Loop i
13. End Loop j
14. i=0
15. Repeat Step 16 until (!ax[i]<x)
16. x0=mx[i]
17. sum=0
18. y0=my[i]

19.fun=1

20.p=(x–x0)/h

21.sum=y0

22.k=1; k<=4; k++

23.fun=(fun*(p–(k–1)))/k

24.sum=sum+fun*diff[i][k]

25.End loop k

26.Print Output x,sum

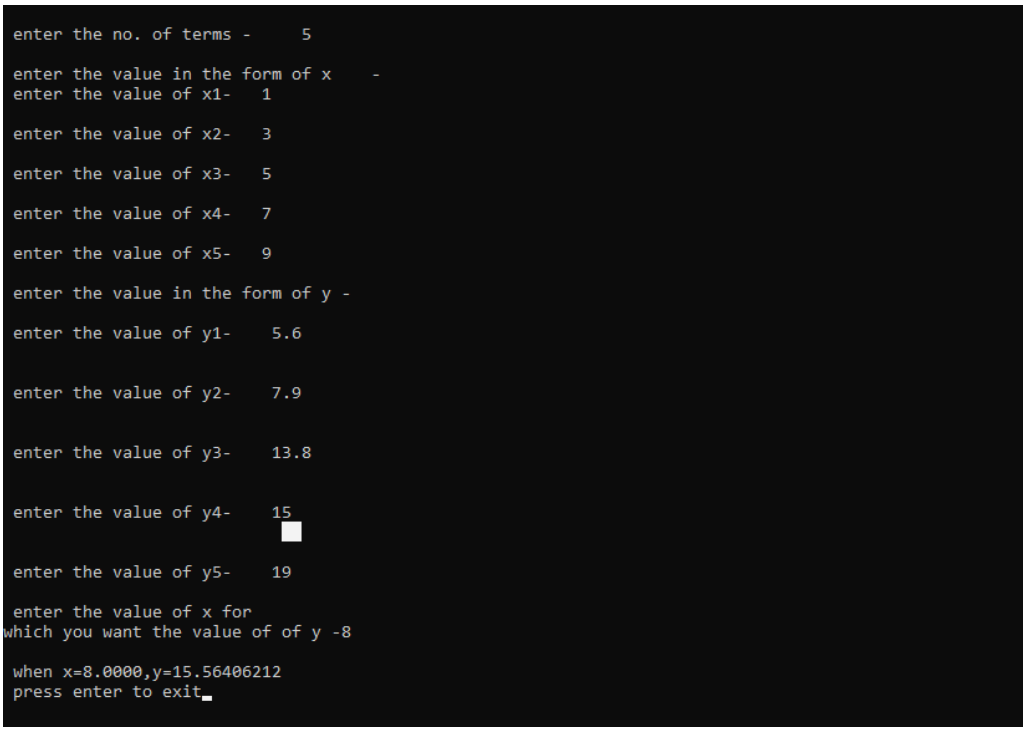
27.End of Program

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
#include<string.h>
void main()
{
int n,i,j,k;
float mx[10],my[10],x,x0=0,y0,sum,h,fun,p,diff[20][20],y1,y2,y3,y4;
clrscr();
printf("\n enter the no. of terms -     ");
scanf("%d",&n);
printf("\n enter the value in the form of x    -  ");
for(i=0;i<n;i++)
{
printf("\n enter the value of x%d-   ",i+1);
scanf("%f",&mx[i]);
}
printf("\n enter the value in the form of y -    ");
for(i=0;i<n;i++)
{
printf("\n\n enter the value of y%d-    ",i+1);
scanf("%f",&my[i]);
}
printf("\n enter the value of x for");
printf("\nwhich you want the value of of y -");

scanf("%f",&x);h=mx[1]-mx[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=my[i+1]-my[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
while(!mx[i]>x)
{
 i++;
```

```
}
x0=mx[i];
sum=0;
y0=my[i];
fun=1;
p=(x-x0)/h;
sum=y0;
for(k=1;k<=4;k++)
{
fun=(fun*(p-(k-1))/k);
sum=sum+fun*diff[i][k];}
printf("\n when x=%6.4f,y=%6.8f",x,sum);
printf("\n press enter to exit");
getch();
}
```

Output



Program 13

Objective: Program to implement Lagrange's Interpolation Method.

Algorithm:

1.Start

2. Read number of data (n)

3. Read data Xi and Yi for i=1 ton n

4. Read value of independent variables say xp whose corresponding value of dependent say yp is to be determined.

5. Initialize: yp = 0

6. For i = 1 to n
   Set p = 1
   For j =1 to n
   If i ≠ j then
   Calculate p = p * (xp - Xj)/(Xi - Xj)
   End If
   Next j
   Calculate yp = yp + p * Yi
   Next i

7. Display value of yp as interpolated value.

8.Stop

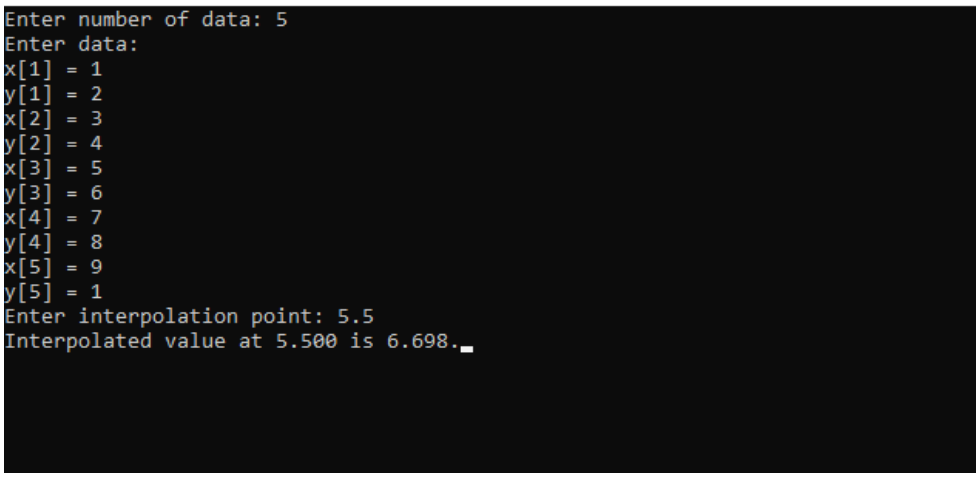
Source Code

```
#include<stdio.h>
```

```
#include<conio.h>

void main()
{
    float x[100], y[100], xp, yp=0, p;
    int i,j,n;
    clrscr();
    /* Input Section */
    printf("Enter number of data: ");
    scanf("%d", &n);
    printf("Enter data:\n");
    for(i=1;i<=n;i++)
    {
        printf("x[%d] = ", i);
        scanf("%f", &x[i]);
        printf("y[%d] = ", i);
        scanf("%f", &y[i]);
    }
    printf("Enter interpolation point: ");
    scanf("%f", &xp);
    /* Implementing Lagrange Interpolation */
    for(i=1;i<=n;i++)
    {
        p=1;
        for(j=1;j<=n;j++)
        {
            if(i!=j)
            {
                p = p* (xp - x[j])/(x[i] - x[j]);
            }
        }
        yp = yp + p * y[i];
```

```
    }
    printf("Interpolated value at %.3f is %.3f.", xp, yp);
    getch();
}
```

Output



Program 14

Objective: Program to implement Trapezoidal rule.

Algorithm:

- Start
- Define and Declare function
- Input initial boundary value, final boundary value and length of interval
- Calculate number of strips, n = (final boundary value –final boundary value)/length of interval
- Perform following operation in loop

  x[i]=x0+i*h

  y[i]=f(x[i])

  print y[i]

- Initialize se=0, s0=0
- Do the following using loop

  If i %2 = 0

  So=s0+y[i]

  Otherwise

  Se=se+y[i]

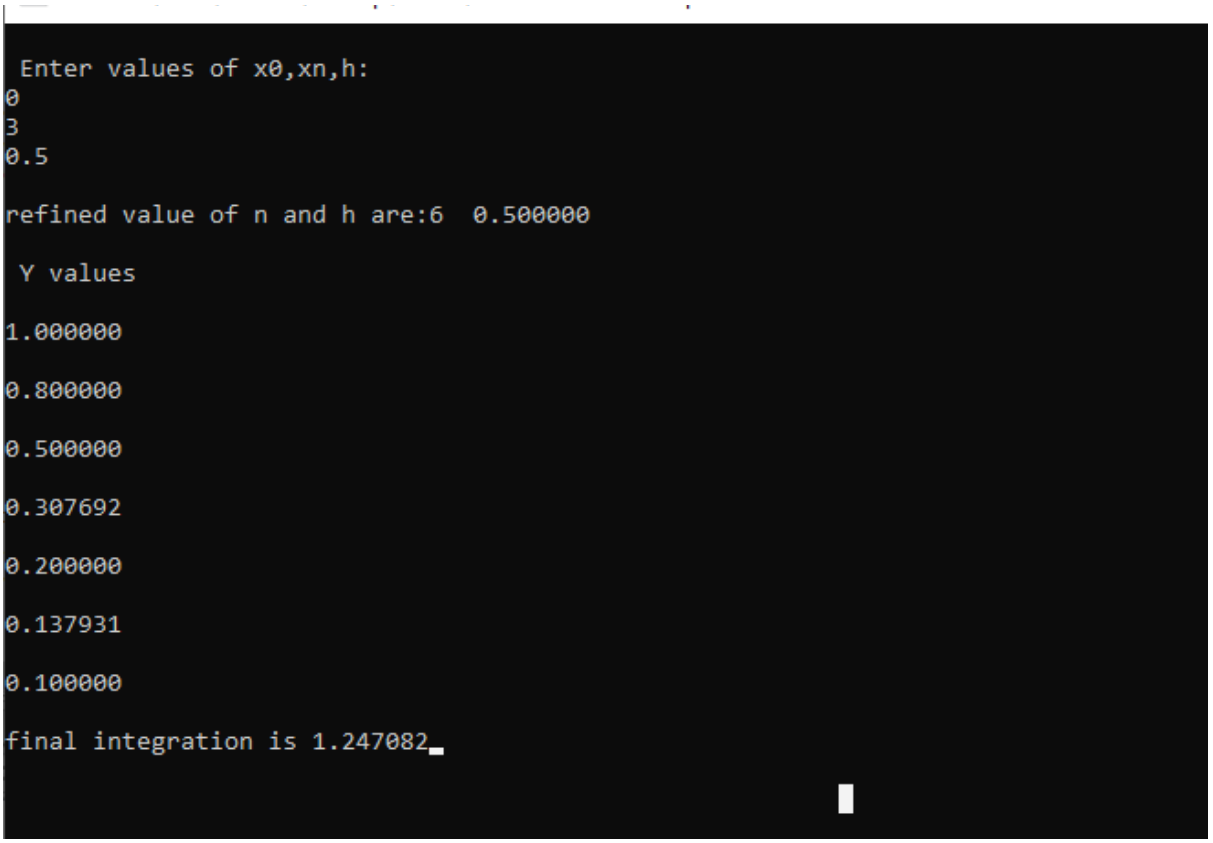
- ans= h/3*(y[0]+y[n]+4*so+2*se)
- print the ans
- Stop

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(float x)
```

```
{
   return(1/(1+pow(x,2)));
}
void main(){
   int i,n;
   float x0,xn,h,y[20],so,se,ans,x[20];
   printf("\n Enter values of x0,xn,h:\n");
   scanf("%f%f%f",&x0,&xn,&h);
   n=(xn-x0)/h;
   if(n%2==1)
   {
      n=n+1;
   }
   h=(xn-x0)/n;
   printf("\nrefined value of n and h are:%d  %f\n",n,h);
   printf("\n Y values \n");
   for(i=0; i<=n; i++)
   {
      x[i]=x0+i*h;
      y[i]=f(x[i]);
      printf("\n%f\n",y[i]);
   }
   so=0;
   se=0;
   for(i=1; i<n; i++)
   {
      if(i%2==1)
      {
         so=so+y[i];
      }
      else
      {
         se=se+y[i];
      }
   }
   ans=h/3*(y[0]+y[n]+4*so+2*se);
   printf("\nfinal integration is %f",ans);
   getch();
}
```

Output



Program 15

Objective: Program to implement Simpson 1/3 rule.

Algorithm:

Step 1. Start;

Step 2. Input function f(x);

Step 3. Read a,b,n; // the lower and upper limits and number of sub-intervals

Step 4. Compute h=(b-a)/n;

Step 5. Sum = [f(a)-f(a+nh)];

Step 6. for i=1 to n-1 step 2 do

Compute sum = sum + 4*f(a+ih)+2*f(a+(i+1)h);

endfor;

Step 7. Compute result = sum * h/3;

Step 8. Print result;

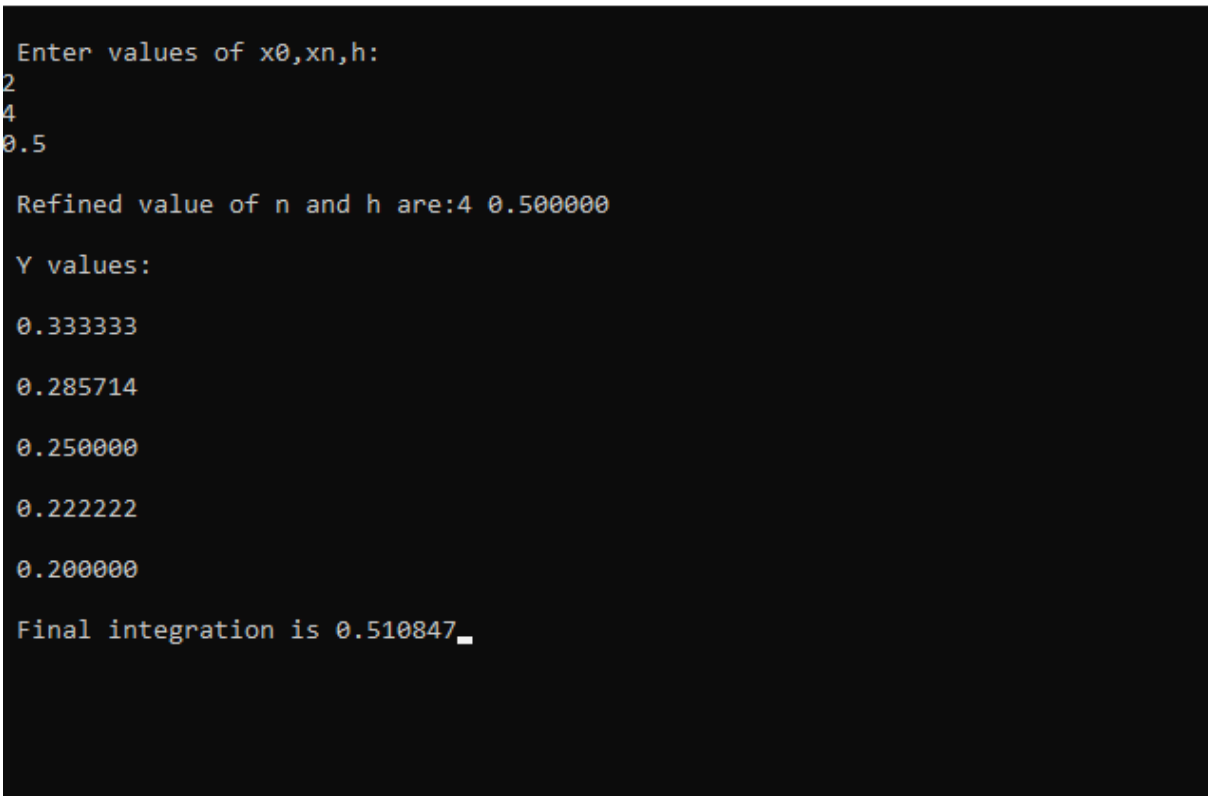
Step 9. Stop;

Source Code

```
#include<stdio.h>
#include<conio.h>
float f(float x) {
```

```
    return(1/(1+x));
}
void main() {
    int i,n;
    float x0,xn,h,y[20],so,se,ans,x[20];
    printf("\n Enter values of x0,xn,h: ");
    scanf("%f%f%f",&x0,&xn,&h);
    n=(xn-x0)/h;
    if(n%2==1)
    {
        n=n+1;
    }
    h=(xn-x0)/n;
    printf("\n Refined value of n and h are:%d %f\n",n,h);
    printf("\n Y values: \n");
    for(i=0; i<=n; i++)
    {
        x[i]=x0+i*h;
        y[i]=f(x[i]);
        printf("\n %f\n",y[i]);
    }
    so=0;
    se=0;
    for(i=1; i<n; i++)
    {
        if(i%2==1)
        {
            so=so+y[i];
        }
        else
        {
            se=se+y[i];
        }

    }
    ans=h/3*(y[0]+y[n]+4*so+2*se);
    printf("\n Final integration is %f",ans);

    getch();
}
```

Output



Program 16

Objective: Program to implement Simpson 3/8 rule.

Algorithm:

1.Given a function f(x):

2. (Get user inputs) Input a,b=endpoints of interval n=number of intervals

3. Set h= (b-a)/n.

4. Set sum=0.

5. Begin For i= 1 to n -1 Set x =a + h*i. If i%3=0 Then Set sum=sum+2*f(x) Else Set sum=sum+3*f(x) End For

6. Set sum = sum + f(a)+f(b)

7. Set ans = sum*(3h/8)

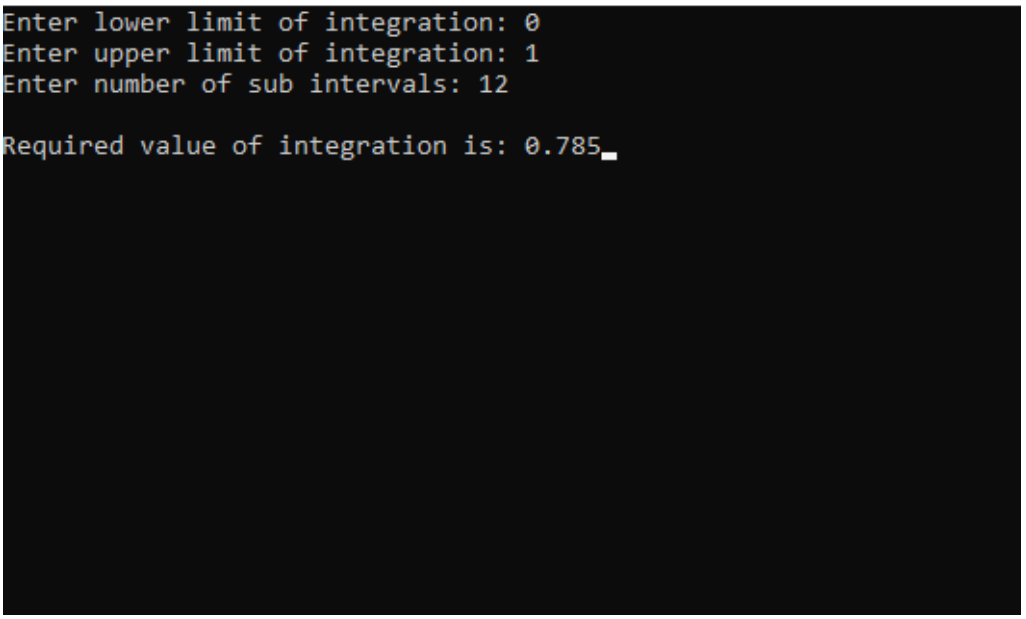
8.End

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x) 1/(1+x*x)
int main() {
 float lower, upper, integration=0.0, stepSize, k;
 int i, subInterval;
```

```
 printf("Enter lower limit of integration: ");
 scanf("%f", &lower);
 printf("Enter upper limit of integration: ");
 scanf("%f", &upper);
 printf("Enter number of sub intervals: ");
 scanf("%d", &subInterval);
stepSize = (upper - lower)/subInterval;
integration = f(lower) + f(upper);
 for(i=1; i<= subInterval-1; i++)
 {
  k = lower + i*stepSize;
  if(i%3 == 0)
  {
   integration = integration + 2 * f(k);
  }
  else
  {
   integration = integration + 3 * f(k);
  } }
 integration = integration * stepSize*3/8;
 printf("\nRequired value of integration is: %.3f", integration);
 getch();
 return 0;
}
```

Output



Program 17

Objective: Program to Fit a line.

Algorithm:

1. Take two floating types of array. x[10] and y[10].

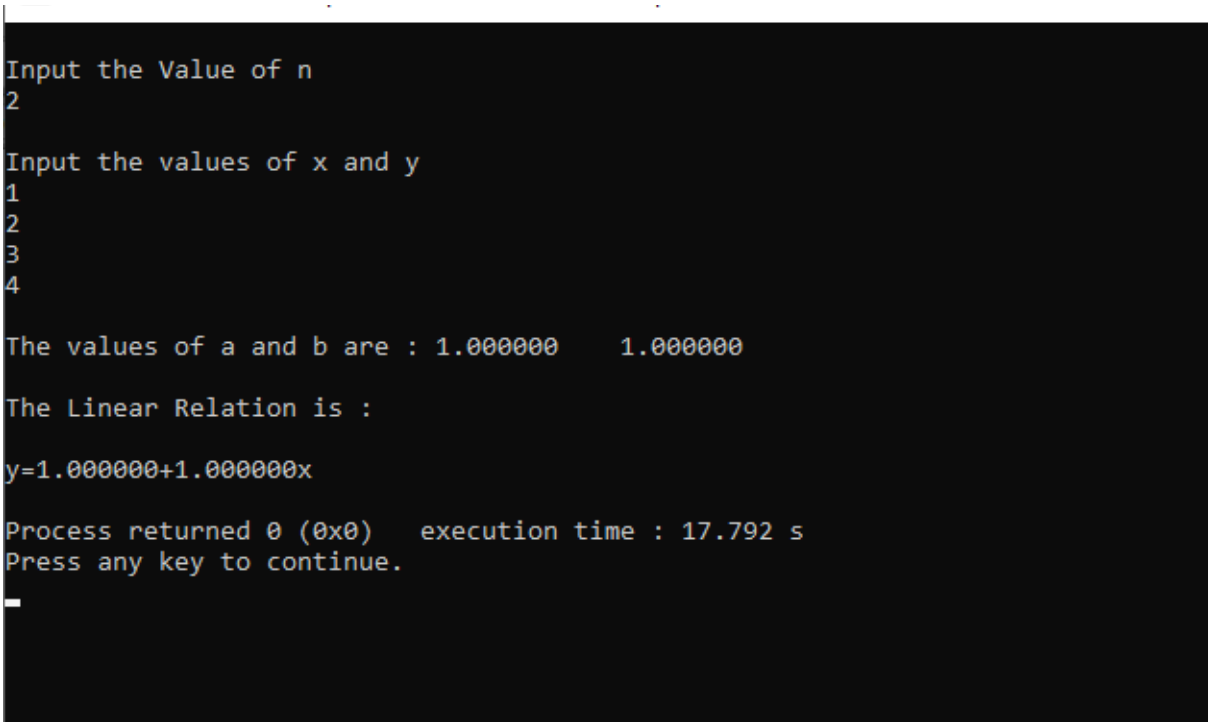
2. Now take n as input from the user.
3. Take x,y as input from the user with the help of for loop. (0<=i<n)
4. Now find the values of Σx, Σy, Σxy, Σx^2 using for loop.
5. Using the formula to solve the simultaneous equation solve for a and b. For formula have a look at Fig 2.
6. Put the values of a and b in the equation of the straight line by checking the condition if b>0 or b=0.
7. Print the result.

Source Code

```
#include<stdio.h>
int main() {
    int i,j,n;
    float x[10],y[10],sum1=0,sum2=0,sum3=0,sum4=0;
```

```
    float a,b,d;
    printf("\nInput the Value of n\n");
    scanf("%d",&n);
    printf("\nInput the values of x and y\n");
    for(i=0;i<n;i++)  {
      scanf("%f%f",&x[i],&y[i]);
    }
    for(i=0;i<n;i++)
    {
      sum1=sum1+x[i];
      sum2=sum2+x[i]*x[i];
      sum3=sum3+y[i];
      sum4=sum4+x[i]*y[i];
    }
    d=n*sum2-sum1*sum1;
    a=(sum2*sum3-sum1*sum4)/d;
    b=(n*sum4-sum1*sum3)/d;
    printf("\nThe values of a and b are : %f\t%f\n",a,b);
    printf("\nThe Linear Relation is : \n");
    if(b>0) {
     printf("\ny=%f+%fx\n",a,b);
    }
    else
    {
     printf("y=%f%fx",a,b);
    }
}
```

Output



Program 18

Objective: Program to Fit a Parabola.

Algorithm:

1. Read no. of data points n and order of polynomial Mp .
2. Read data values .
3. If n< Mp [Regression is not possible] stop else continue ;

4. Set M=Mp + 1 ;
5. Compute co-efficient of C-matrix .
6. Compute co-efficient of B-matrix .
7. Solve for the co-efficients a1,a2,. . . . . an .
8. Write the co-efficient .
9. Estimate the function value at the glren of independents variables .

Source Code

```
#include<stdio.h>
#include<math.h>
int main(){
   float xy[20][20], matrix[3][4], ratio, a;
   float sum_x = 0, sum_y = 0, sum_x2 = 0, sum_x3 = 0, sum_x4
=0, sum_xy = 0, sum_x2y = 0;
   int i, j , k, n;
   printf("Enter no of data: ");
   scanf("%d", &n);
   printf("Enter the data: \n");
```

```
for(i = 0; i < 2; i++){
    for(j = 0; j < n; j++){
        scanf("%f", &xy[i][j]);
    }
}
for(i = 0; i < n; i++){
    sum_x += xy[0][i];
    sum_y += xy[1][i];
    sum_x2 += pow(xy[0][i], 2);
    sum_x3 += pow(xy[0][i], 3);
    sum_x4 += pow(xy[0][i], 4);
    sum_xy += xy[0][i]*xy[1][i];
    sum_x2y += pow(xy[0][i], 2) * xy[1][i];
}
matrix[0][0] = n;
matrix[0][1] = sum_x;
matrix[0][2] = sum_x2;
matrix[0][3] = sum_y;
matrix[1][0] = sum_x;
matrix[1][1] = sum_x2;
matrix[1][2] = sum_x3;
matrix[1][3] = sum_xy;
matrix[2][0] = sum_x2;
matrix[2][1] = sum_x3;
matrix[2][2] = sum_x4;
matrix[2][3] = sum_x2y;
for(i = 0; i < 3; i++){
    for(j = 0; j < 3; j++){
        if(i!=j){
            ratio = matrix[j][i]/matrix[i][i];
            for(k = 0; k < 4; k++){
                matrix[j][k] -= ratio * matrix[i][k];
```

```
            }
        }
    }
    for(i = 0; i < 3; i++){
        a = matrix[i][i];
        for(j = 0; j < 4; j++){
            matrix[i][j] /= a;
        }
    }
    for(i = 0; i < 3; i++){
        printf("\n%c => %.2f", 97+i, matrix[i][3]);
    }
}
```

Output

```
Enter no of data: 6
Enter the data:
1
2
3
5
5
4
6
7
8
9
4
5

a => 5.50
b => 0.97
c => -0.17
Process returned 0 (0x0)   execution time : 19.232 s
Press any key to continue.
```