Graphic Era
HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

Term work

on

Computer Based Numerical and Statistical Techniques

(PMA 402)

2020-21

Submitted to:

Mr. Sumeshwar

Assistant Professor
GEHU, D.Dun

Submitted by:

Shivam Patwal

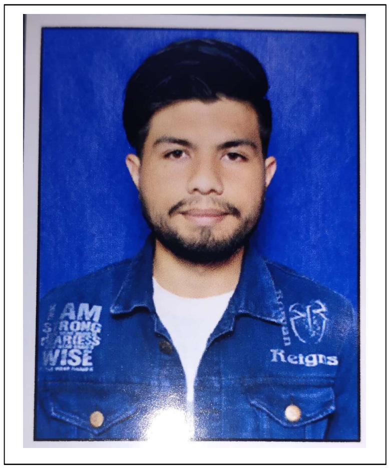
University Roll. No.: 1918694
Class Roll. No./Section: 35/J

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

ACKNOWLEDGMENT

I would like to particularly thank my CBNST Lab Faculty Mr. Sumeshwar 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.



Shivam Patwal

Uni. Roll No.-1918694

B.Tech CSE-J-IV Sem

Session: 2020-2021

GEHU, Dehradun

Patwal

Graphic Era
HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

Table of Contents

Program 1

Objective: Program to find Errors

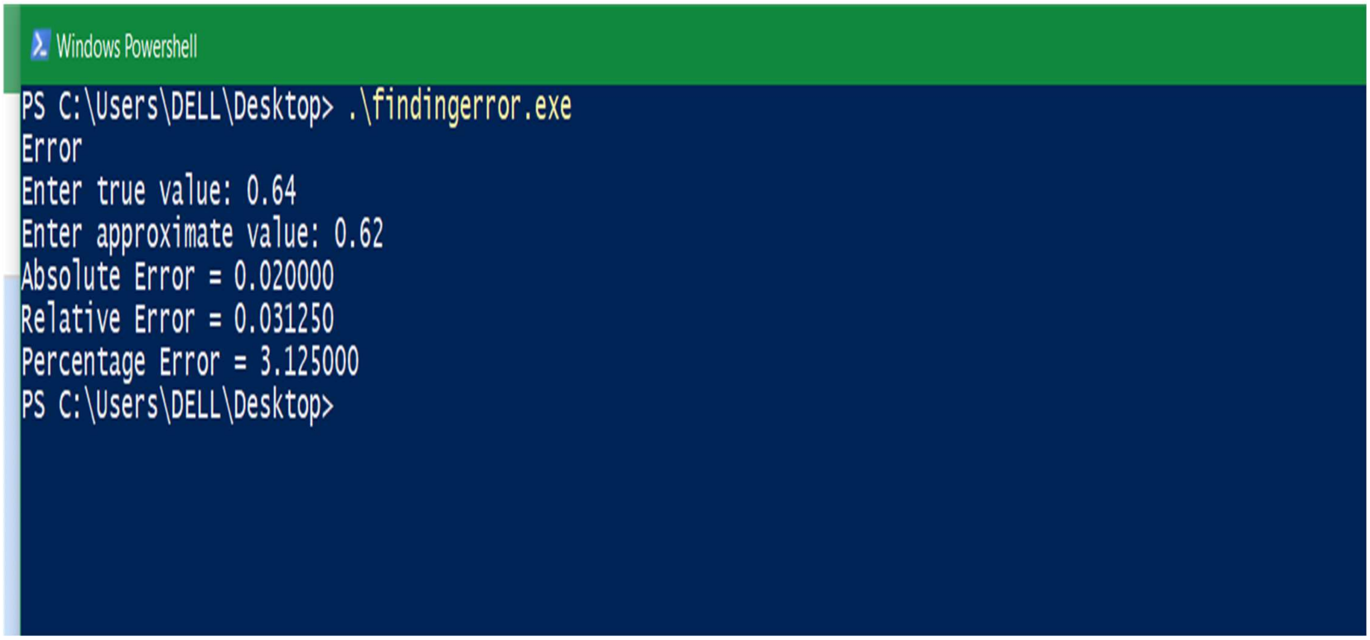
Algorithm:

```
1. start
2. enter true and approximate value.
3. Absolute Error=true value - approximate value.
4. Relative Error=approximate value / true value.
5. Percentage Error=Relative Error * 100.
6. Display Absolute Error, Relative Error, Percentage Error.
7. Stop
```

Source Code

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

int main()
{
   printf("Error\n");
   double relativeE, absoluteE, percentageE, trueV, approxV;
   printf("Enter true value: ");
   scanf("%lf", &trueV);
   printf("Enter approximate value: ");
   scanf("%lf", &approxV);
   absoluteE = trueV-approxV;
   relativeE = absoluteE/trueV;
   percentageE = relativeE*100;
   printf("Absolute Error = %lf\n",fabs(absoluteE));
   printf("Relative Error = %lf\n",fabs(relativeE));
   printf("Percentage Error = %lf",fabs(percentageE));
}
```

output



```
PS C:\Users\DELL\Desktop> .\findingerror.exe
Error
Enter true value: 0.64
Enter approximate value: 0.62
Absolute Error = 0.020000
Relative Error = 0.031250
Percentage Error = 3.125000
PS C:\Users\DELL\Desktop>
```

Program 2

Objective: Program to Implement Bisection Method

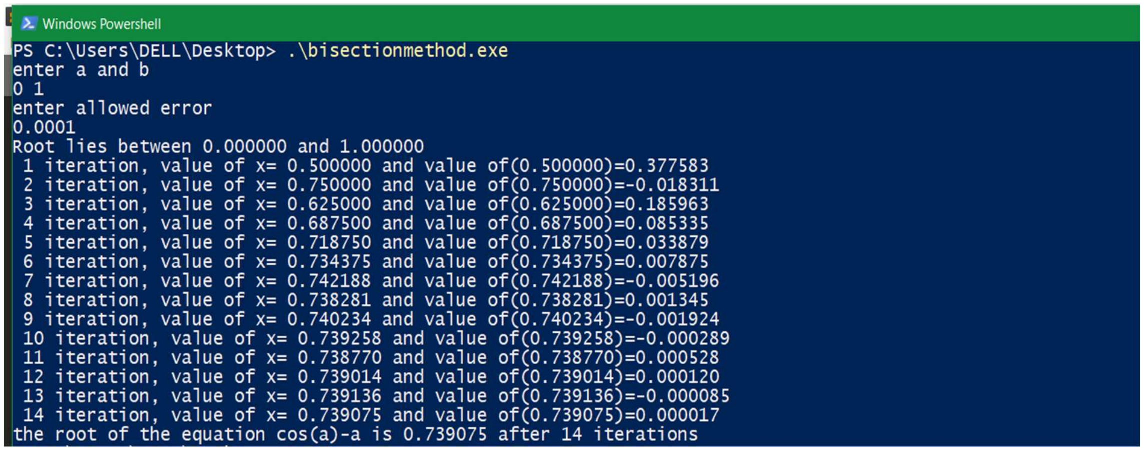
Algorithm:

```
1. start
2. Define function f(x)
3. Choose initial guesses x0 and x1 such that f(x0)f(x1) < 0
4. Choose pre-specified tolerable error e.
5. Calculate new approximated root as x2 = (x0 + x1)/2
6. Calculate f(x0)f(x2)
        a. if f(x0)f(x2) < 0 then x0 = x0 and x1 = x2
        b. if f(x0)f(x2) > 0 then x0 = x2 and x1 = x1
        c. if f(x0)f(x2) = 0 then goto (8)
7. if |f(x2)| > e then goto (5) otherwise goto (8)
8. Display x2 as root.
9. Stop
```

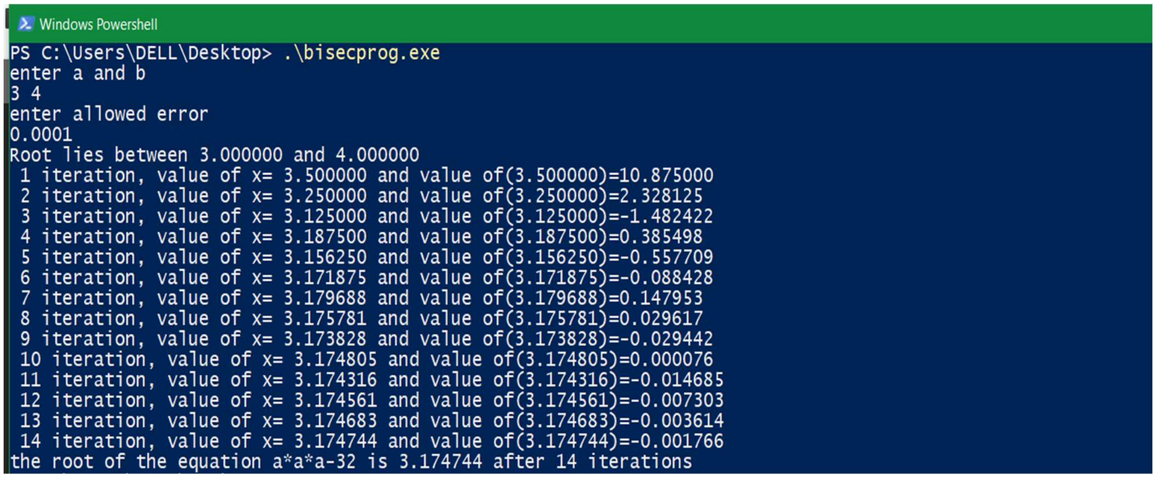
Source Code

```
#include<stdio.h>
#include<math.h>
float fun(float a){
    return (a*a*a-32);
}

int main(){
  float a,b,mid,error;
  int cond=0,itr=0;
do{  printf("enter a and b \n");
scanf("%f %f",&a,&b);
printf("enter allowed error\n");
scanf("%f",&error);
if(fun(a)*fun(b)<0) {
    printf("Root lies between %f and %f \n",a,b);
cond=1;
while(1){
     mid=(a+b)/(float)2;
if(fabs(b-a)<error) break;
itr++;
if(fun(a)*fun(mid)<0) {   b=mid;     }
if(fun(mid)*fun(b)<0) {   a=mid;     }
printf(" %d iteration, value of x= %f and value of(%f)=%f\n",itr,mid,mid,fun(mid));
}
printf("the root of the equation  is %f after %d iterations\n",a,itr);
}else {
    printf("invalid input!! \n enter again \n");
}
}while(cond==0);

return 0;
}
```

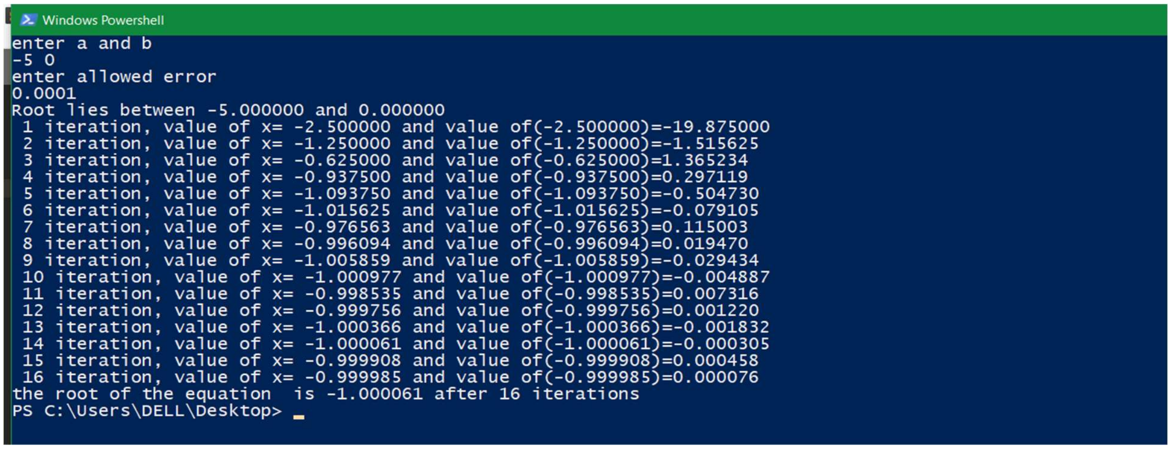
Output



```
PS C:\Users\DELL\Desktop> .\bisectionmethod.exe
enter a and b
0 1
enter allowed error
0.0001
Root lies between 0.000000 and 1.000000
 1 iteration, value of x= 0.500000 and value of(0.500000)=0.377583
 2 iteration, value of x= 0.750000 and value of(0.750000)=-0.018311
 3 iteration, value of x= 0.625000 and value of(0.625000)=0.185963
 4 iteration, value of x= 0.687500 and value of(0.687500)=0.085335
 5 iteration, value of x= 0.718750 and value of(0.718750)=0.033879
 6 iteration, value of x= 0.734375 and value of(0.734375)=0.007875
 7 iteration, value of x= 0.742188 and value of(0.742188)=-0.005196
 8 iteration, value of x= 0.738281 and value of(0.738281)=0.001345
 9 iteration, value of x= 0.740234 and value of(0.740234)=-0.001924
 10 iteration, value of x= 0.739258 and value of(0.739258)=-0.000289
 11 iteration, value of x= 0.738770 and value of(0.738770)=0.000528
 12 iteration, value of x= 0.739014 and value of(0.739014)=0.000120
 13 iteration, value of x= 0.739136 and value of(0.739136)=-0.000085
 14 iteration, value of x= 0.739075 and value of(0.739075)=0.000017
the root of the equation cos(a)-a is 0.739075 after 14 iterations
```



```
PS C:\Users\DELL\Desktop> .\bisecprog.exe
enter a and b
3 4
enter allowed error
0.0001
Root lies between 3.000000 and 4.000000
 1 iteration, value of x= 3.500000 and value of(3.500000)=10.875000
 2 iteration, value of x= 3.250000 and value of(3.250000)=2.328125
 3 iteration, value of x= 3.125000 and value of(3.125000)=-1.482422
 4 iteration, value of x= 3.187500 and value of(3.187500)=0.385498
 5 iteration, value of x= 3.156250 and value of(3.156250)=-0.557709
 6 iteration, value of x= 3.171875 and value of(3.171875)=-0.088428
 7 iteration, value of x= 3.179688 and value of(3.179688)=0.147953
 8 iteration, value of x= 3.175781 and value of(3.175781)=0.029617
 9 iteration, value of x= 3.173828 and value of(3.173828)=-0.029442
 10 iteration, value of x= 3.174805 and value of(3.174805)=0.000076
 11 iteration, value of x= 3.174316 and value of(3.174316)=-0.014685
 12 iteration, value of x= 3.174561 and value of(3.174561)=-0.007303
 13 iteration, value of x= 3.174683 and value of(3.174683)=-0.003614
 14 iteration, value of x= 3.174744 and value of(3.174744)=-0.001766
the root of the equation a*a*a-32 is 3.174744 after 14 iterations
```



```
enter a and b
-5 0
enter allowed error
0.0001
Root lies between -5.000000 and 0.000000
 1 iteration, value of x= -2.500000 and value of(-2.500000)=-19.875000
 2 iteration, value of x= -1.250000 and value of(-1.250000)=-1.515625
 3 iteration, value of x= -0.625000 and value of(-0.625000)=1.365234
 4 iteration, value of x= -0.937500 and value of(-0.937500)=0.297119
 5 iteration, value of x= -1.093750 and value of(-1.093750)=-0.504730
 6 iteration, value of x= -1.015625 and value of(-1.015625)=-0.079105
 7 iteration, value of x= -0.976563 and value of(-0.976563)=0.115003
 8 iteration, value of x= -0.996094 and value of(-0.996094)=0.019470
 9 iteration, value of x= -1.005859 and value of(-1.005859)=-0.029434
 10 iteration, value of x= -1.000977 and value of(-1.000977)=-0.004887
 11 iteration, value of x= -0.998535 and value of(-0.998535)=0.007316
 12 iteration, value of x= -0.999756 and value of(-0.999756)=0.001220
 13 iteration, value of x= -1.000366 and value of(-1.000366)=-0.001832
 14 iteration, value of x= -1.000061 and value of(-1.000061)=-0.000305
 15 iteration, value of x= -0.999908 and value of(-0.999908)=0.000458
 16 iteration, value of x= -0.999985 and value of(-0.999985)=0.000076
the root of the equation  is -1.000061 after 16 iterations
PS C:\Users\DELL\Desktop> _
```

Program 3

Objective: Program to Implement Regula Falsi Method

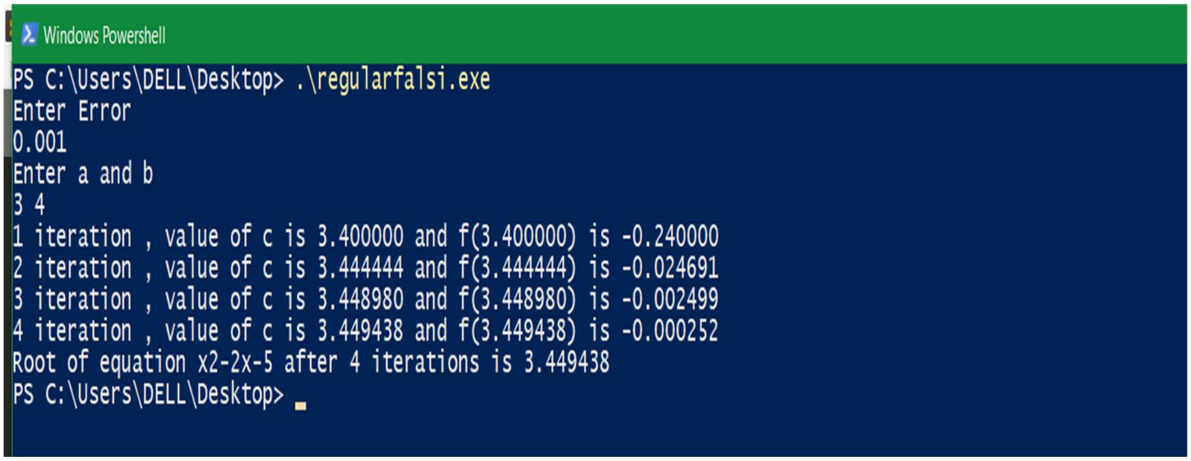
ALGORITHM-

```
1. start
2. Define function f(x)
3. Choose initial guesses x0 and x1 such that f(x0)f(x1) < 0
4. Choose pre-specified tolerable error e.
5. Calculate new approximated root as:
   x2 = x0 - ((x0-x1) * f(x0))/(f(x0) - f(x1))
6. Calculate f(x0)f(x2)
        a. if f(x0)f(x2) < 0 then x0 = x0 and x1 = x2
        b. if f(x0)f(x2) > 0 then x0 = x2 and x1 = x1
        c. if f(x0)f(x2) = 0 then goto (8)
7. if |f(x2)|>e then goto (5) otherwise goto (8)
8. Display x2 as root.
9. Stop
```

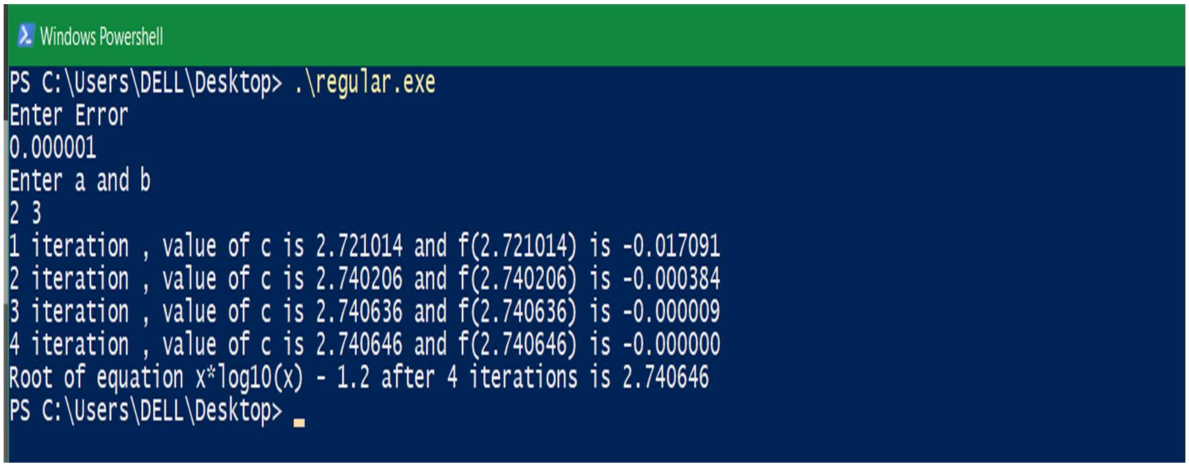
Source Code

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

double fun(double x)
{
 return x*x-2*x-5;
}
int main()
{
   double a,b,err,c;
   printf("Enter Error\n");
   scanf("%lf",&err);
   bool input=true;
  do
 {
   printf("Enter a and b\n");
   scanf("%lf%lf",&a,&b);
   if(fun(a)*fun(b)>0)
  printf("Invalid Input Try Again!!\n");
else
{
     input=false;
     int it=1;
while(1)
{
    c= a- (b-a)*fun(a)/(double)(fun(b)-fun(a));
    if(fun(c)*fun(a)<0) b=c;
    else a=c;
   printf("%d iteration , value of c is %lf and f(%lf) is %lf\n",it,c,c,fun(c));
   if(fabs(fun(c))<=err) break;
       it++;
 }
    printf("Root of equation x^2-2x-5 after %d iterations is %lf\n",it,c);
 }
 } while(input);
}
```

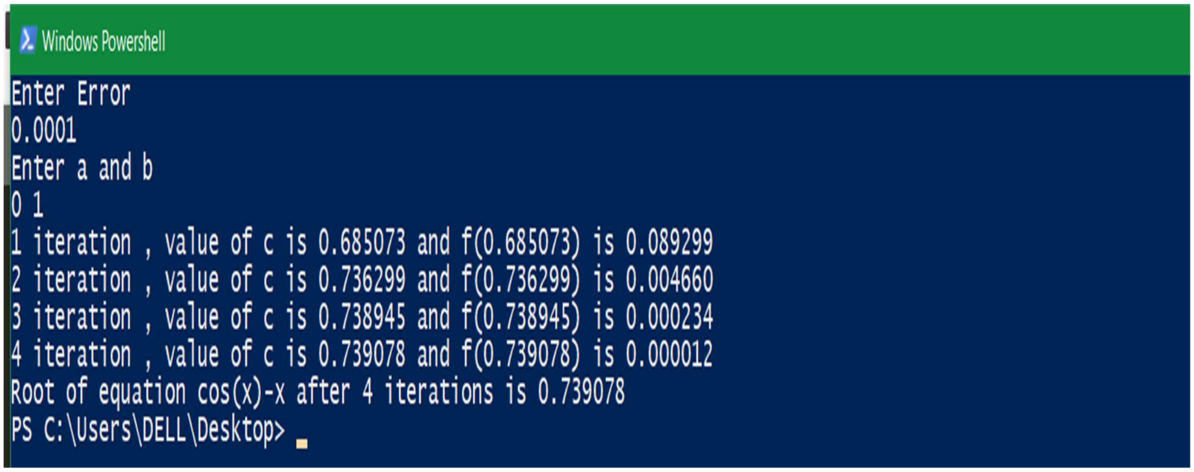
Output



```
PS C:\Users\DELL\Desktop> .\regularfalsi.exe
Enter Error
0.001
Enter a and b
3 4
1 iteration , value of c is 3.400000 and f(3.400000) is -0.240000
2 iteration , value of c is 3.444444 and f(3.444444) is -0.024691
3 iteration , value of c is 3.448980 and f(3.448980) is -0.002499
4 iteration , value of c is 3.449438 and f(3.449438) is -0.000252
Root of equation x2-2x-5 after 4 iterations is 3.449438
PS C:\Users\DELL\Desktop> _
```



```
PS C:\Users\DELL\Desktop> .\regular.exe
Enter Error
0.000001
Enter a and b
2 3
1 iteration , value of c is 2.721014 and f(2.721014) is -0.017091
2 iteration , value of c is 2.740206 and f(2.740206) is -0.000384
3 iteration , value of c is 2.740636 and f(2.740636) is -0.000009
4 iteration , value of c is 2.740646 and f(2.740646) is -0.000000
Root of equation x*log10(x) - 1.2 after 4 iterations is 2.740646
PS C:\Users\DELL\Desktop> _
```



```
Enter Error
0.0001
Enter a and b
0 1
1 iteration , value of c is 0.685073 and f(0.685073) is 0.089299
2 iteration , value of c is 0.736299 and f(0.736299) is 0.004660
3 iteration , value of c is 0.738945 and f(0.738945) is 0.000234
4 iteration , value of c is 0.739078 and f(0.739078) is 0.000012
Root of equation cos(x)-x after 4 iterations is 0.739078
PS C:\Users\DELL\Desktop> _
```

Program 4

Objective: Program to Implement Secant Method

ALGORITHM-

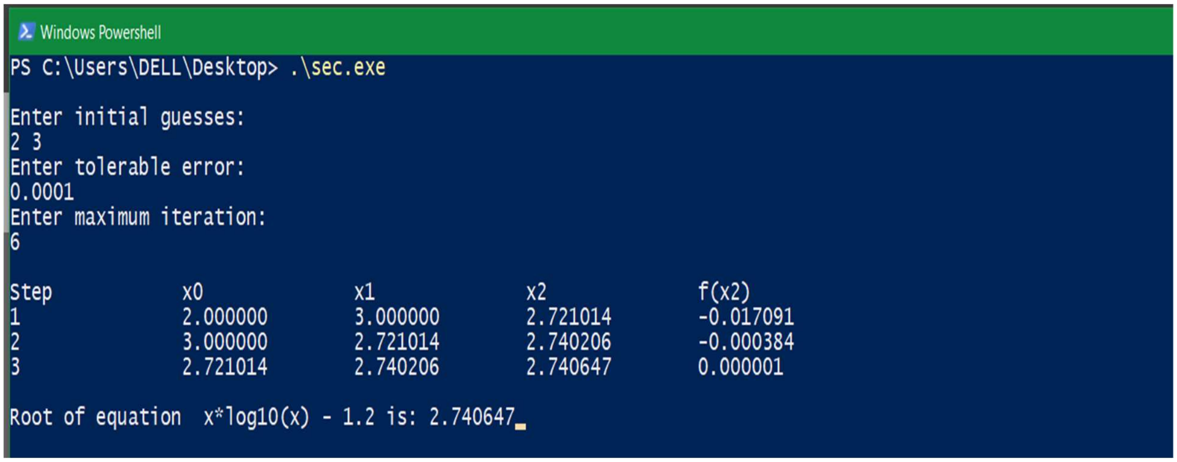
```
1. Start
2. Define function as f(x)
3. Input initial guesses (x0 and x1),
   tolerable error (e) and maximum iteration (N)
4. Initialize iteration counter i = 1
5. If f(x0) = f(x1) then print "Mathematical Error"
   and goto (11) otherwise goto (6)
6. Calcualte x2 = x1 - (x1-x0) * f(x1) / ( f(x1) - f(x0) )
7. Increment iteration counter i = i + 1
8. If i >= N then print "Not Convergent"
   and goto (11) otherwise goto (9)
9. If |f(x2)| > e then set x0 = x1, x1 = x2
   and goto (5) otherwise goto (10)
10. Print root as x2
11. Stop
```

Source Code

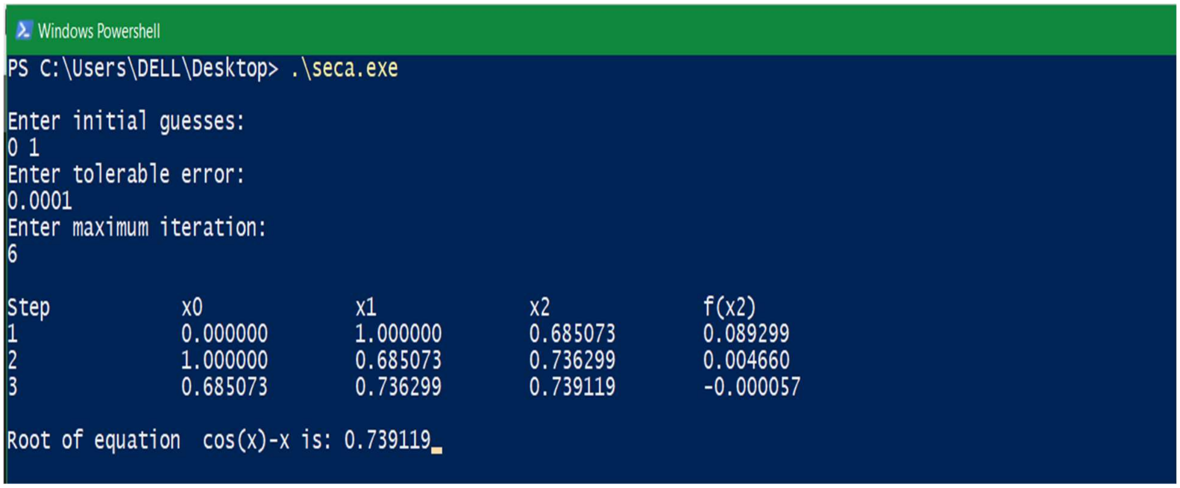
```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
#define    f(x)    x*x*x - 2*x - 5

int main(){
        float x0, x1, x2, f0, f1, f2, e;
        int step = 1, N;
        clrscr();
        printf("\nEnter initial guesses:\n");
        scanf("%f%f", &x0, &x1);
        printf("Enter tolerable error:\n");
        scanf("%f", &e);
        printf("Enter maximum iteration:\n");
        scanf("%d", &N);
        printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n");
        do{
                f0 = f(x0);
                f1 = f(x1);
                if(f0 == f1) {
```

```
                printf("Mathematical Error.");
                exit(0);
            }
            x2 = x1 - (x1 - x0) * f1/(f1-f0);
            f2 = f(x2);
            printf("%d\t\t%f\t%f\t%f\t%f\n",step,x0,x1,x2, f2);
            x0 = x1;
            f0 = f1;
            x1 = x2;
            f1 = f2;
            step = step + 1;
            if(step > N){
                printf("Not Convergent.");
                exit(0);
            }
        }while(fabs(f2)>e);

        printf("\nRoot is: %f", x2);
        getch();
}
```

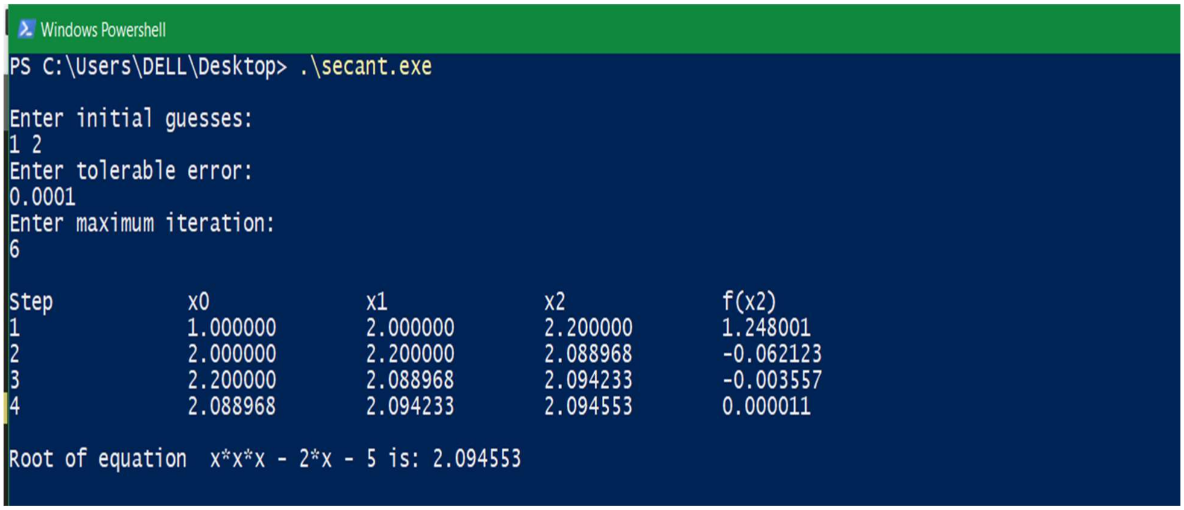
Output



```
PS C:\Users\DELL\Desktop> .\sec.exe

Enter initial guesses:
2 3
Enter tolerable error:
0.0001
Enter maximum iteration:
6

Step            x0              x1              x2              f(x2)
1               2.000000        3.000000        2.721014        -0.017091
2               3.000000        2.721014        2.740206        -0.000384
3               2.721014        2.740206        2.740647        0.000001

Root of equation  x*log10(x) - 1.2 is: 2.740647_
```



```
PS C:\Users\DELL\Desktop> .\seca.exe

Enter initial guesses:
0 1
Enter tolerable error:
0.0001
Enter maximum iteration:
6

Step            x0              x1              x2              f(x2)
1               0.000000        1.000000        0.685073        0.089299
2               1.000000        0.685073        0.736299        0.004660
3               0.685073        0.736299        0.739119        -0.000057

Root of equation  cos(x)-x is: 0.739119_
```



```
PS C:\Users\DELL\Desktop> .\secant.exe

Enter initial guesses:
1 2
Enter tolerable error:
0.0001
Enter maximum iteration:
6

Step            x0              x1              x2              f(x2)
1               1.000000        2.000000        2.200000        1.248001
2               2.000000        2.200000        2.088968        -0.062123
3               2.200000        2.088968        2.094233        -0.003557
4               2.088968        2.094233        2.094553        0.000011

Root of equation  x*x*x - 2*x - 5 is: 2.094553
```

Program 5

Objective: Program to Implement Iteration Method

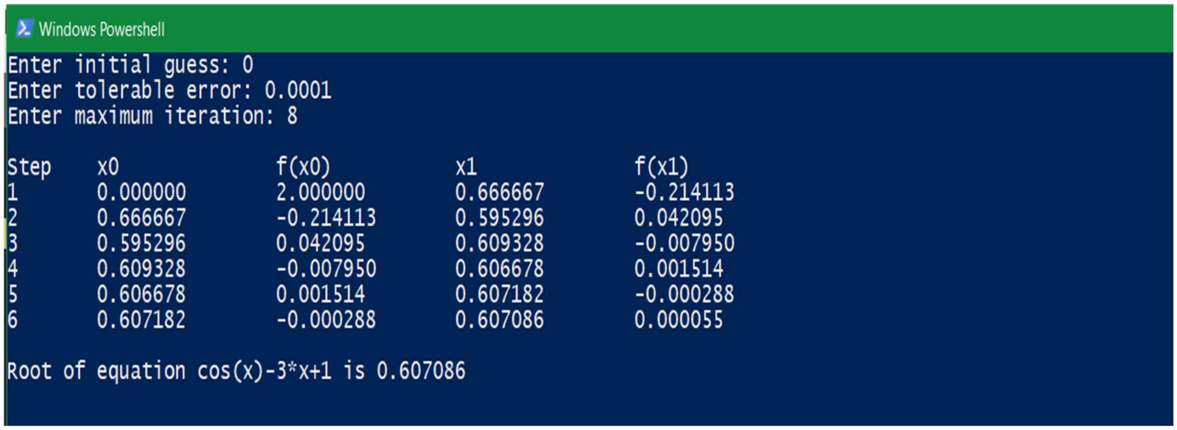
ALOGORITHM-

```
1. Start
2. Define function f(x)
3. Define function g(x) which is obtained
   from f(x)=0 such that x = g(x) and |g'(x) < 1|
4. Choose intial guess x0, Tolerable Error e
   and Maximum Iteration N
5. Initialize iteration counter: step = 1
6. Calculate x1 = g(x0)
7. Increment iteration counter: step = step + 1
8. If step > N then print "Not Convergent"
   and goto (12) otherwise goto (10)
9. Set x0 = x1 for next iteration
10. If |f(x1)| > e then goto step (6) otherwise goto step (11)
11. Display x1 as root.
12. Stop
```

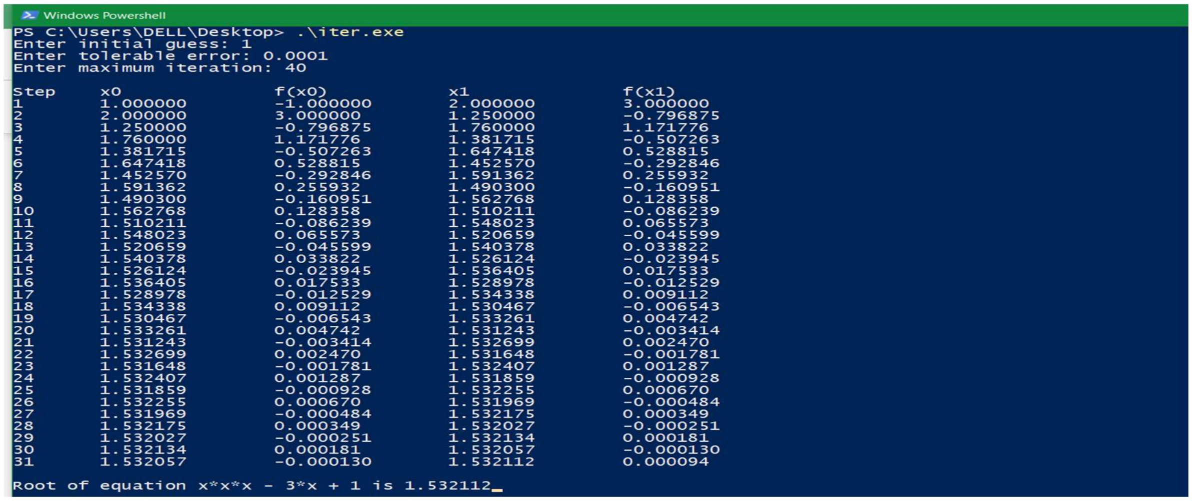
Source code

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

#define   f(x)   cos(x)-3*x+1
#define   g(x)   (1+cos(x))/3

int main(){
        int step=1, N;
        float x0, x1, e;
        clrscr();
        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);
}
```

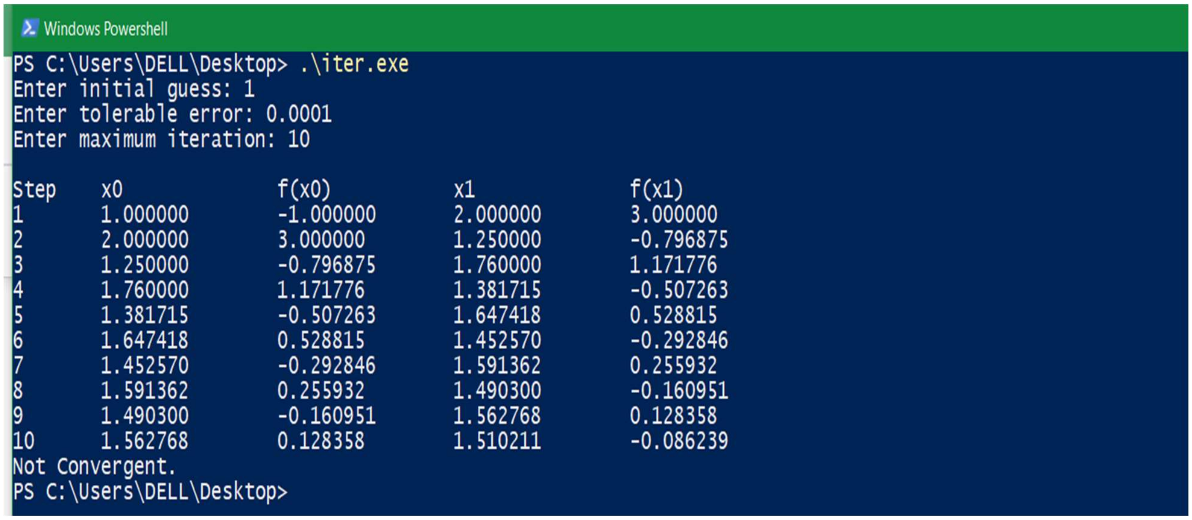
Output



```
Enter initial guess: 0
Enter tolerable error: 0.0001
Enter maximum iteration: 8

Step    x0              f(x0)           x1              f(x1)
1       0.000000        2.000000        0.666667        -0.214113
2       0.666667        -0.214113       0.595296        0.042095
3       0.595296        0.042095        0.609328        -0.007950
4       0.609328        -0.007950       0.606678        0.001514
5       0.606678        0.001514        0.607182        -0.000288
6       0.607182        -0.000288       0.607086        0.000055

Root of equation cos(x)-3*x+1 is 0.607086
```



```
PS C:\Users\DELL\Desktop> .\iter.exe
Enter initial guess: 1
Enter tolerable error: 0.0001
Enter maximum iteration: 40

Step    x0              f(x0)           x1              f(x1)
1       1.000000        -1.000000       2.000000        3.000000
2       2.000000        3.000000        1.250000        -0.796875
3       1.250000        -0.796875       1.760000        1.171776
4       1.760000        1.171776        1.381715        -0.507263
5       1.381715        -0.507263       1.647418        0.528815
6       1.647418        0.528815        1.452570        -0.292846
7       1.452570        -0.292846       1.591362        0.255932
8       1.591362        0.255932        1.490300        -0.160951
9       1.490300        -0.160951       1.562768        0.128358
10      1.562768        0.128358        1.510211        -0.086239
11      1.510211        -0.086239       1.548023        0.065573
12      1.548023        0.065573        1.520659        -0.045599
13      1.520659        -0.045599       1.540378        0.033822
14      1.540378        0.033822        1.526124        -0.023945
15      1.526124        -0.023945       1.536405        0.017533
16      1.536405        0.017533        1.528978        -0.012529
17      1.528978        -0.012529       1.534338        0.009112
18      1.534338        0.009112        1.530467        -0.006543
19      1.530467        -0.006543       1.533261        0.004742
20      1.533261        0.004742        1.531243        -0.003414
21      1.531243        -0.003414       1.532699        0.002470
22      1.532699        0.002470        1.531648        -0.001781
23      1.531648        -0.001781       1.532407        0.001287
24      1.532407        0.001287        1.531859        -0.000928
25      1.531859        -0.000928       1.532255        0.000670
26      1.532255        0.000670        1.531969        -0.000484
27      1.531969        -0.000484       1.532175        0.000349
28      1.532175        0.000349        1.532027        -0.000251
29      1.532027        -0.000251       1.532134        0.000181
30      1.532134        0.000181        1.532057        -0.000130
31      1.532057        -0.000130       1.532112        0.000094

Root of equation x*x*x - 3*x + 1 is 1.532112_
```



```
PS C:\Users\DELL\Desktop> .\iter.exe
Enter initial guess: 1
Enter tolerable error: 0.0001
Enter maximum iteration: 10

Step    x0              f(x0)           x1              f(x1)
1       1.000000        -1.000000       2.000000        3.000000
2       2.000000        3.000000        1.250000        -0.796875
3       1.250000        -0.796875       1.760000        1.171776
4       1.760000        1.171776        1.381715        -0.507263
5       1.381715        -0.507263       1.647418        0.528815
6       1.647418        0.528815        1.452570        -0.292846
7       1.452570        -0.292846       1.591362        0.255932
8       1.591362        0.255932        1.490300        -0.160951
9       1.490300        -0.160951       1.562768        0.128358
10      1.562768        0.128358        1.510211        -0.086239
Not Convergent.
PS C:\Users\DELL\Desktop>
```

Program 6

Objective: Program to Implement Newton Raphson Method

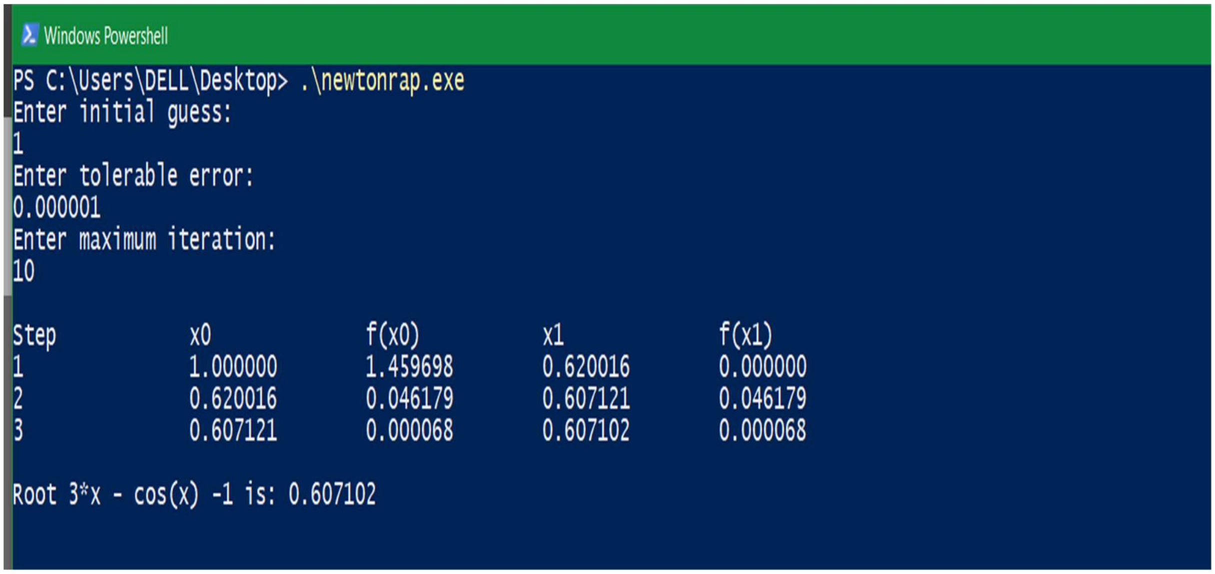
ALGORITHM-

```
1. Start
2. Define function as f(x)
3. Define first derivative of f(x) as g(x)
4. Input initial guess (x0), tolerable error (e)
   and maximum iteration (N)
5. Initialize iteration counter i = 1
6. If g(x0) = 0 then print "Mathematical Error"
   and goto (12) otherwise goto (7)
7. Calcualte x1 = x0 - f(x0) / g(x0)
8. Increment iteration counter i = i + 1
9. If i >= N then print "Not Convergent"
   and goto (12) otherwise goto (10)
10. If |f(x1)| > e then set x0 = x1
    and goto (6) otherwise goto (11)
11. Print root as x1
12. Stop
```

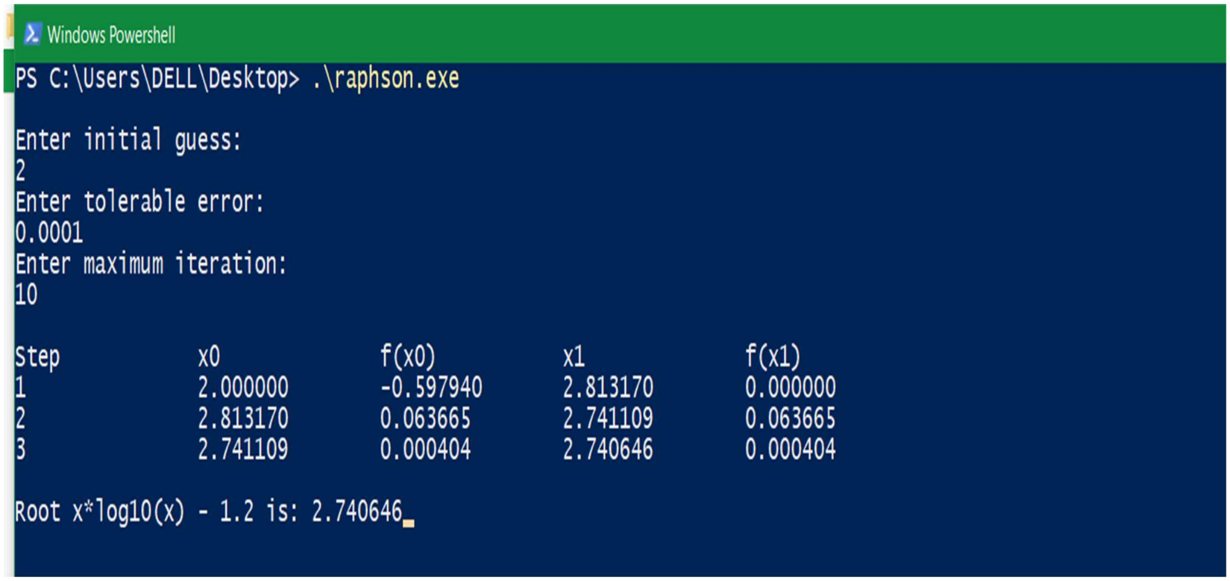
Source Code

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

void main(){
        float x0, x1, f0, f1, g0, e;
        int step = 1, N;
        clrscr();
        printf("\nEnter initial guess:\n");
        scanf("%f", &x0);
        printf("Enter tolerable error:\n");
        scanf("%f", &e);
        printf("Enter maximum iteration:\n");
        scanf("%d", &N);
        printf("\nStep\t\tx0\t\tf(x0)\t\tx1\t\tf(x1)\n");
        do {
                g0 = g(x0);
                f0 = f(x0);
        if(g0 == 0.0){
            printf("Mathematical Error.");
            exit(0);
        }
        x1 = x0 - f0/g0;
        printf("%d\t\t%f\t%f\t%f\t%f\n",step,x0,f0,x1,f1);
        x0 = x1;
        step = step+1;
        if(step > N) {
            printf("Not Convergent.");
            exit(0);
        }
        f1 = f(x1);

    }while(fabs(f1)>e);

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

Output



```
PS C:\Users\DELL\Desktop> .\newtonrap.exe
Enter initial guess:
1
Enter tolerable error:
0.000001
Enter maximum iteration:
10

Step            x0              f(x0)           x1              f(x1)
1               1.000000        1.459698        0.620016        0.000000
2               0.620016        0.046179        0.607121        0.046179
3               0.607121        0.000068        0.607102        0.000068

Root 3*x - cos(x) -1 is: 0.607102
```



```
PS C:\Users\DELL\Desktop> .\raphson.exe

Enter initial guess:
2
Enter tolerable error:
0.0001
Enter maximum iteration:
10

Step            x0              f(x0)           x1              f(x1)
1               2.000000        -0.597940       2.813170        0.000000
2               2.813170        0.063665        2.741109        0.063665
3               2.741109        0.000404        2.740646        0.000404

Root x*log10(x) - 1.2 is: 2.740646
```

Program 7

Objective: Program to Implement Gauss Elimination Method

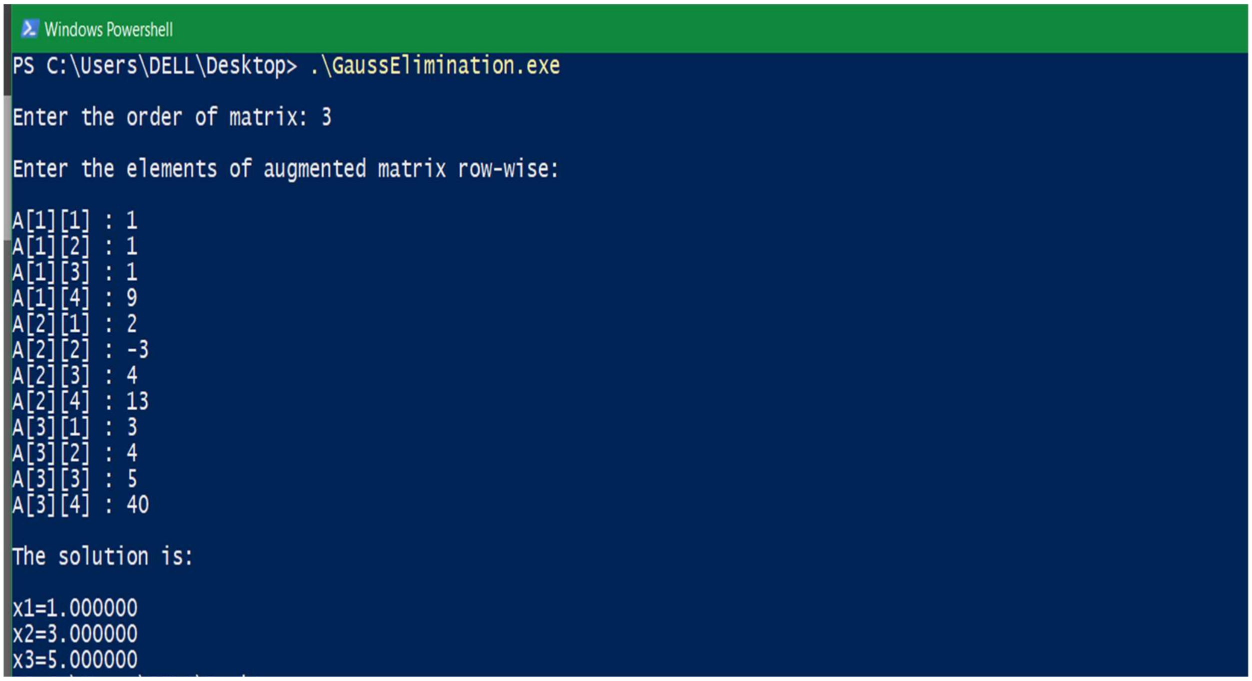
ALGORITHM-

```
1. Start
2. Read Number of Unknowns: n
3. Read Augmented Matrix (A) of n by n+1 Size
4. Transform Augmented Matrix (A)
   to Upper Trainagular Matrix by Row Operations.
5. Obtain Solution by Back Substitution.
6. Display Result.
7. 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];
    /* this loop is for backward substitution*/
    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]); /* x1, x2, x3 are the required solutions*/
    }
    return(0);
}
```

Output



```
PS C:\Users\DELL\Desktop> .\GaussElimination.exe

Enter the order of matrix: 3

Enter the elements of augmented matrix row-wise:

A[1][1] : 1
A[1][2] : 1
A[1][3] : 1
A[1][4] : 9
A[2][1] : 2
A[2][2] : -3
A[2][3] : 4
A[2][4] : 13
A[3][1] : 3
A[3][2] : 4
A[3][3] : 5
A[3][4] : 40

The solution is:

x1=1.000000
x2=3.000000
x3=5.000000
```

Program 8

Objective: Program to Implement Gauss Jordan Method

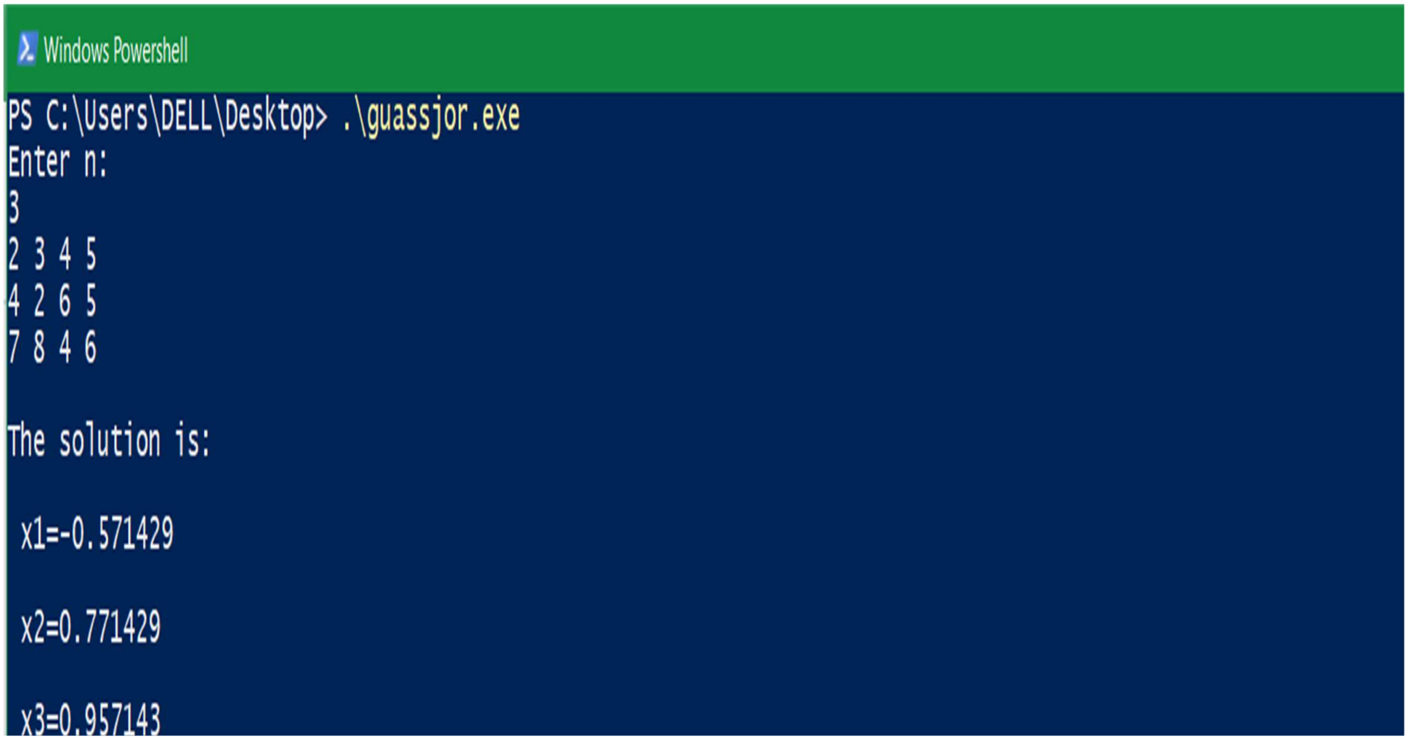
ALGORITHM-

```
1. Start
2. Read Number of Unknowns: n
3. Read Augmented Matrix (A) of n by n+1 Size
4. Transform Augmented Matrix (A)
   to Diagonal Matrix by Row Operations.
5. Obtain Solution by Making All Diagonal Elements to 1.
6. Display Result.
7. Stop
```

Source code

```
#include<stdio.h>
int main(){
int i,j,k,n;
printf("Enter n:\n");
scanf("%d",&n);
float arr[n][n+1],c,x[10];
for(i=1;i<=n;i++){
for(j=1;j<=n+1;j++){
scanf("%f",&arr[i][j]);
}
}

for(j=1;j<=n;j++){
for(i=1;i<=n;i++){
if(j!=i){
c=arr[i][j]/arr[i][i];

for(k=1;k<=n+1;k++){
arr[i][k]=arr[i][k]-c*arr[j][k];}
}
}
}
 printf("\nThe solution is:\n");
    for(i=1; i<=n; i++)
    {
        x[i]=arr[i][n+1]/arr[i][i];
        printf("\n x%d=%f\n",i,x[i]);
    }
}
```

Output



```
PS C:\Users\DELL\Desktop> .\guassjor.exe
Enter n:
3
2 3 4 5
4 2 6 5
7 8 4 6

The solution is:

 x1=-0.571429

 x2=0.771429

 x3=0.957143
```

Program 9

Objective: Program to Implement Gauss Jacobi Method

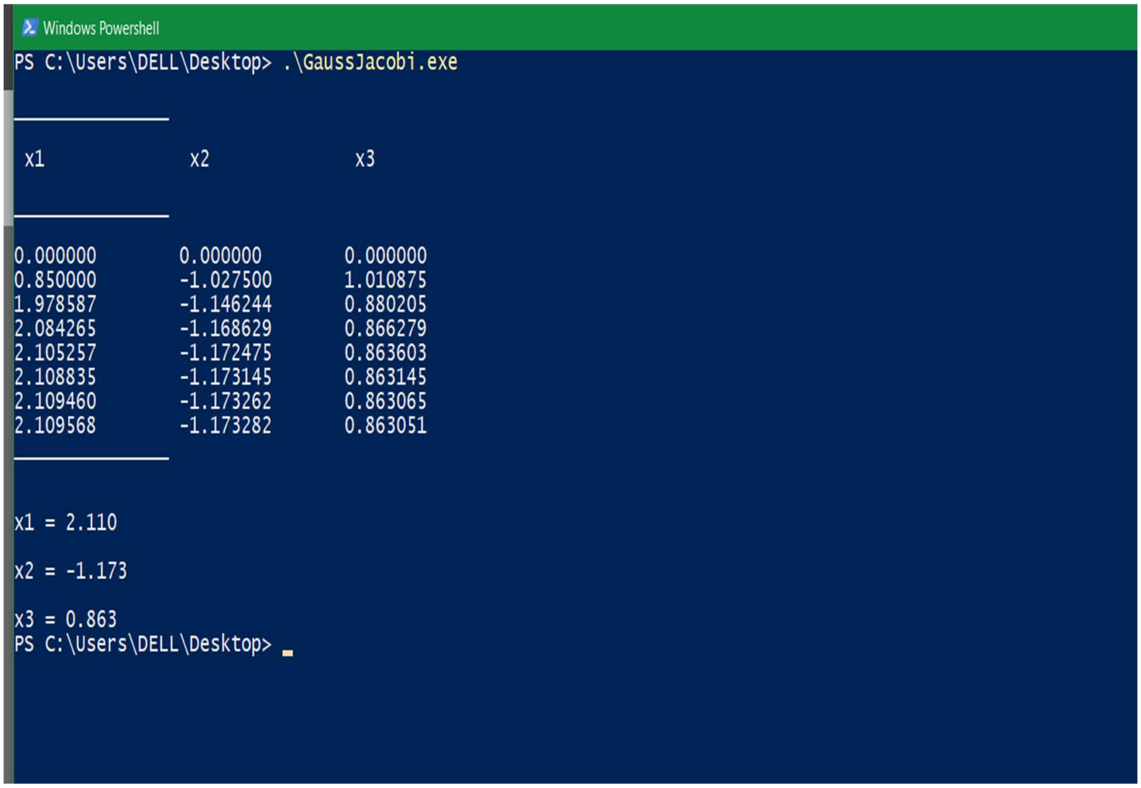
ALGORITHM-

1. Start
2. Arrange given system of linear equations in diagonally dominant form
3. Read tolerable error (e)
4. Convert the first equation in terms of first variable, second equation in terms of second variable and so on.
5. Set initial guesses for x0, y0, z0 and so on
6. Substitute value of x0, y0, z0 ... from step 5 in equation obtained in step 4 to calculate new values x1, y1, z1 and so on
7. If| x0 - x1| > e and | y0 - y1| > e and | z0 - z1| > e and so on then goto step 9
8. Set x0=x1, y0=y1, z0=z1 and so on and goto step 6
9. Print value of x1, y1, z1 and so on
10. Stop

Source Code

```
#include<stdio.h>
#include<math.h>
#define ESP 0.0001
#define X1(x2,x3) ((17 - 20*(x2) + 2*(x3))/20)
#define X2(x1,x3) ((-18 - 3*(x1) + (x3))/20)
#define X3(x1,x2) ((25 - 2*(x1) + 3*(x2))/20)

void main() {
  double x1=0,x2=0,x3=0,y1,y2,y3;
  int i=0;

  printf("\n________________\n");
  printf("\n x1\t\t x2\t\t x3\n");
  printf("\n________________\n");
  printf("\n%f\t%f\t%f",x1,x2,x3);
  do
  {
    y1=X1(x2,x3);
    y2=X2(y1,x3);
    y3=X3(y1,y2);
  if(fabs(y1-x1)<ESP && fabs(y2-x2)<ESP && fabs(y3-x3)<ESP )
  {
    printf("\n__________________\n");
    printf("\n\nx1 = %.3lf",y1);
    printf("\n\nx2 = %.3lf",y2);
    printf("\n\nx3 = %.3lf",y3);
    i = 1;
  }
  else
   {
    x1 = y1;
    x2 = y2;
    x3 = y3;
    printf("\n%f\t%f\t%f",x1,x2,x3);
   }
  }
  while(i != 1);

 }
```

Output



```
PS C:\Users\DELL\Desktop> .\GaussJacobi.exe

_______________

 x1              x2              x3

_______________

0.000000        0.000000        0.000000
0.850000        -1.027500       1.010875
1.978587        -1.146244       0.880205
2.084265        -1.168629       0.866279
2.105257        -1.172475       0.863603
2.108835        -1.173145       0.863145
2.109460        -1.173262       0.863065
2.109568        -1.173282       0.863051
_______________


x1 = 2.110

x2 = -1.173

x3 = 0.863
PS C:\Users\DELL\Desktop> _
```

Program 10

Objective: Program to Implement Gauss Seidel Method

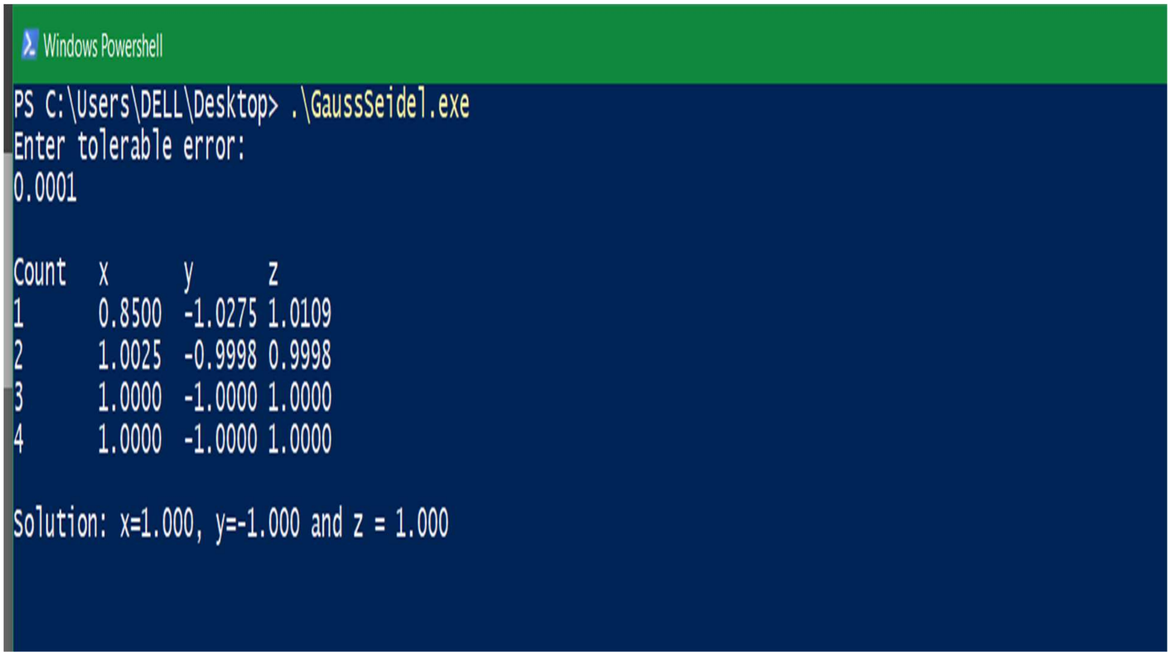
ALGORITHM-

1. Start
2. Arrange given system of linear equations in diagonally dominant form
3. Read tolerable error (e)
4. Convert the first equation in terms of first variable, second equation in terms of second variable and so on.
5. Set initial guesses for x0, y0, z0 and so on
6. Substitute value of y0, z0 ... from step 5 in first equation obtained from step 4 to calculate new value of x1. Use x1, z0, u0 .... in second equation obtained from step 4 to caluclate new value of y1. Similarly, use x1, y1, u0... to find new z1 and so on.
7. If| x0 - x1| > e and | y0 - y1| > e and | z0 - z1| > e and so on then goto step 9
8. Set x0=x1, y0=y1, z0=z1 and so on and goto step 6
9. Print value of x1, y1, z1 and so on
10. Stop

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f1(x,y,z)  (17-y+2*z)/20
#define f2(x,y,z)  (-18-3*x+z)/20
#define f3(x,y,z)  (25-2*x+3*y)/20

int main()
{
 float x0=0, y0=0, z0=0, x1, y1, z1, e1, e2, e3, e;
 int count=1;
 clrscr();
 printf("Enter tolerable error:\n");
 scanf("%f", &e);

 printf("\nCount\tx\ty\tz\n");
 do
 {
  x1 = f1(x0,y0,z0);
  y1 = f2(x1,y0,z0);
  z1 = f3(x1,y1,z0);
  printf("%d\t%0.4f\t%0.4f\t%0.4f\n",count, x1,y1,z1);

  e1 = fabs(x0-x1);
  e2 = fabs(y0-y1);
  e3 = fabs(z0-z1);

  count++;
  x0 = x1;
  y0 = y1;
  z0 = z1;

 }while(e1>e && e2>e && e3>e);
 printf("\nSolution: x=%0.3f, y=%0.3f and z = %0.3f\n",x1,y1,z1);
 getch();
 return 0;
}
```

Output



```
PS C:\Users\DELL\Desktop> .\GaussSeidel.exe
Enter tolerable error:
0.0001

Count   x       y       z
1       0.8500  -1.0275 1.0109
2       1.0025  -0.9998 0.9998
3       1.0000  -1.0000 1.0000
4       1.0000  -1.0000 1.0000

Solution: x=1.000, y=-1.000 and z = 1.000
```

Program 11

Objective: Program to Implement Newton Forward Interpolation Method

ALGORITHM-

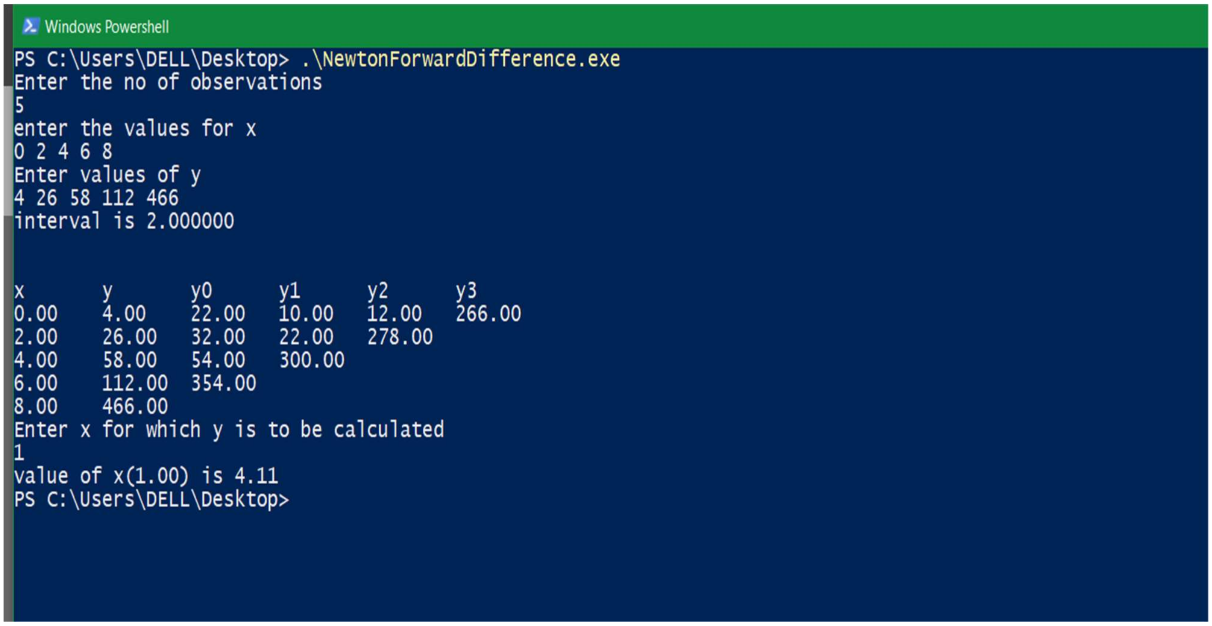
1. Start
2. Enter no of observations : n
3. Enter value of x[20]
4. 3.Enter value of y[20][0]
5. h = x[2]-x[1]
6. enter the value of x for which y is to be calculated. ‘f’
7. u = (f-x[0])/h
8. u1=u
9. a = y[0][0], fact =1

Source Code

```
#include<stdio.h>
int main(){
 int n;
 printf("Enter no of observation: ");
 scanf("%d",&n);
 float x[n],d,y[n][n];
 for(int i=0;i<n;i++){
   scanf("%f",&x[i]);
  }
 for(int i=0;i<n;i++){
    scanf("%f",&y[i][0]);
  }

 printf("Enter vlaue of x for which you want to find value of y: ");
 scanf("%f",&d);
float h,u;
h=x[1]-x[0];
u=(d-x[0])/h;
float a=y[0][0];

 for(int j=1;j<n;j++){
  for(int i=0;i<(n-j) ; i++){
 y[i][j]=y[i+1][j-1]-y[i][j-1];
```

```
   }
  }

for(int i=0;i<n;i++){
printf("%.2f\t",x[i]);
for(int j=0;j< (n-i);j++){

printf("%.2f\t",y[i][j]);

}
printf("\n\n");  }

float fact=1;
float u1=u;
for(int i=1;i<n;i++){
a+=( ( u1*y[0][i])/fact );
u1=u1*(u-(i));
fact=fact*(i+1);
}
printf("value of f(%.2f): %.2f", d,a);
}
```

Output



```
PS C:\Users\DELL\Desktop> .\NewtonForwardDifference.exe
Enter the no of observations
5
enter the values for x
0 2 4 6 8
Enter values of y
4 26 58 112 466
interval is 2.000000


x       y       y0      y1      y2      y3
0.00    4.00    22.00   10.00   12.00   266.00
2.00    26.00   32.00   22.00   278.00
4.00    58.00   54.00   300.00
6.00    112.00  354.00
8.00    466.00
Enter x for which y is to be calculated
1
value of x(1.00) is 4.11
PS C:\Users\DELL\Desktop>
```

Program 12

Objective: Program to Implement Newton Backward Interpolation Method

ALGORITHM-

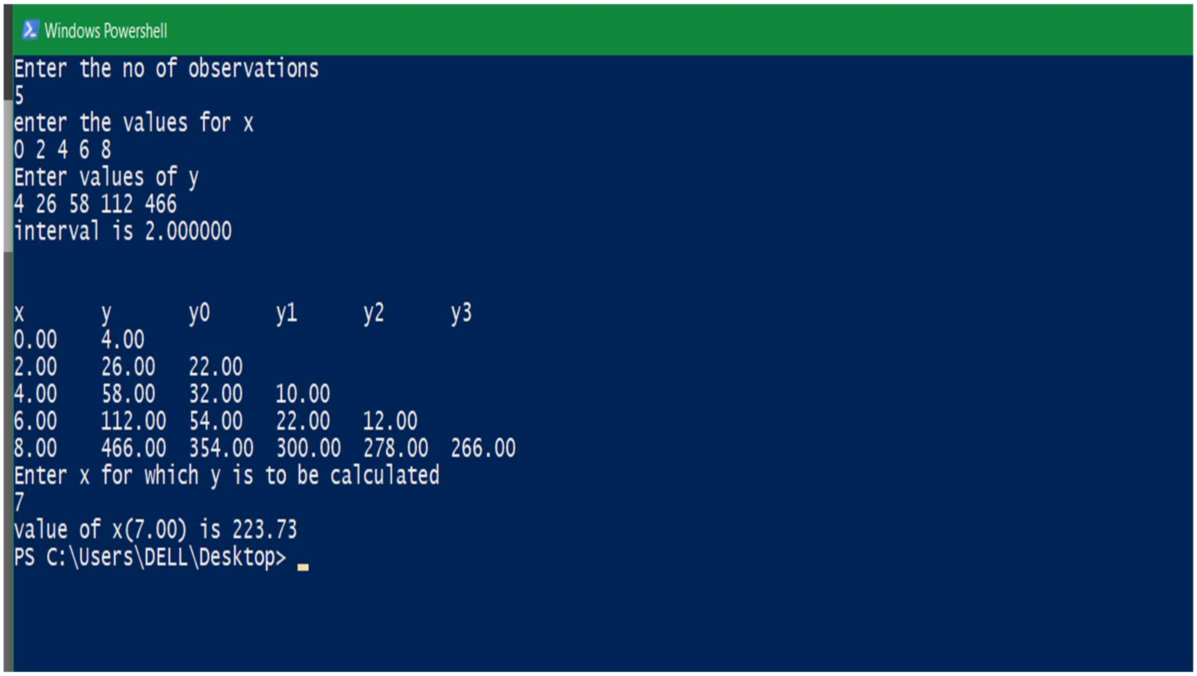
```
1: Enter no of observations : n
2. Enter value of x[20]
3.Enter value of y[20][0]
4. h = x[2]-x[1]
5. enter the value of x for which y is to be calculated. ‘f’
6. u = (f-x[n-1])/h
7. u1=u
8. a = y[n-1][0], fact =1
```

Source Code

```
#include<stdio.h>
int main(){
float h,x[20],y[20][20];
int n;
printf("Enter the no of observations\n");
scanf("%d",&n);
printf("enter the values for x\n");
for(int i=0;i<n;i++)
        scanf("%f",&x[i]);
printf("Enter values of y\n");
for(int i=0;i<n;i++)
        scanf("%f",&y[i][0]);

h = x[2]-x[1];
printf("interval is %f\n",h);

//printf("\tx\ty\n");
//for(int i=1;i<=n;i++)
//      printf("\t%.2f\t%.2f\n",x[i],y[i][1]);
//diff table
printf("\n\n");
for(int i=1;i<n;i++){
        for(int j=n-1;j>i-1;j--){
                y[j][i]=y[j][i-1] - y[j-1][i-1];
        }
}
printf("x\ty\t");
for(int i=0;i<n-1;i++)
```

```
        printf("y%d\t",i);
printf("\n");
for(int i=0;i<n;i++){
        printf("%0.2f",x[i]);
        for(int j=0;j<=i;j++)
                printf("\t%0.2f",y[i][j]);
        printf("\n");
}
float f;
printf("Enter x for which y is to be calculated\n");
scanf("%f",&f);
float u,u1;
u=(f-x[n-1])/h;
u1=u;
float a=y[n-1][0],fact=1;

for(int j=1;j<=n;j++){
        fact=fact*j;
        a=a+u1*y[n-1][j]/fact;
        u1=u1*(u+(j));

}
printf("value of x(%.2f) is %.2f",f,a);
return 0;
}
```

Output



```
Enter the no of observations
5
enter the values for x
0 2 4 6 8
Enter values of y
4 26 58 112 466
interval is 2.000000


x       y       y0      y1      y2      y3
0.00    4.00
2.00    26.00   22.00
4.00    58.00   32.00   10.00
6.00    112.00  54.00   22.00   12.00
8.00    466.00  354.00  300.00  278.00  266.00
Enter x for which y is to be calculated
7
value of x(7.00) is 223.73
PS C:\Users\DELL\Desktop>
```

Program 13

Objective: Program to Implement Lagrange’s Interpolation Method

ALGORITHM-

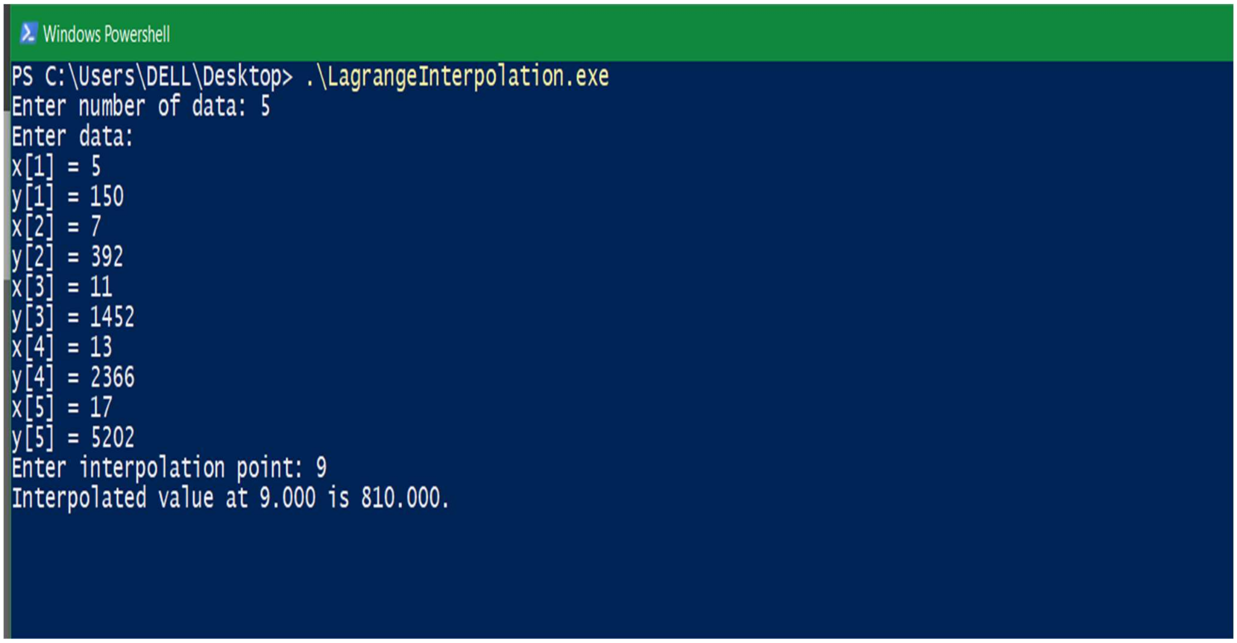
```
1. Start
2. Read number of data (n)
3. Read data Xi and Yi for i=1 ton n
4. Read value of independent variables say xp
   whose corresponding value of dependent say yp is to be determined.
5. Initialize: yp = 0
6. For i = 1 to n
     Set p = 1
     For j =1 to n
       If i ≠ j then
         Calculate p = p * (xp - Xj)/(Xi - Xj)
       End If
     Next j
     Calculate yp = yp + p * Yi
   Next i
6. Display value of yp as interpolated value.
7. 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();
        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);
```

```
            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



```
PS C:\Users\DELL\Desktop> .\LagrangeInterpolation.exe
Enter number of data: 5
Enter data:
x[1] = 5
y[1] = 150
x[2] = 7
y[2] = 392
x[3] = 11
y[3] = 1452
x[4] = 13
y[4] = 2366
x[5] = 17
y[5] = 5202
Enter interpolation point: 9
Interpolated value at 9.000 is 810.000.
```

Program 14

Objective: Program to Implement Trapezoidal Rule

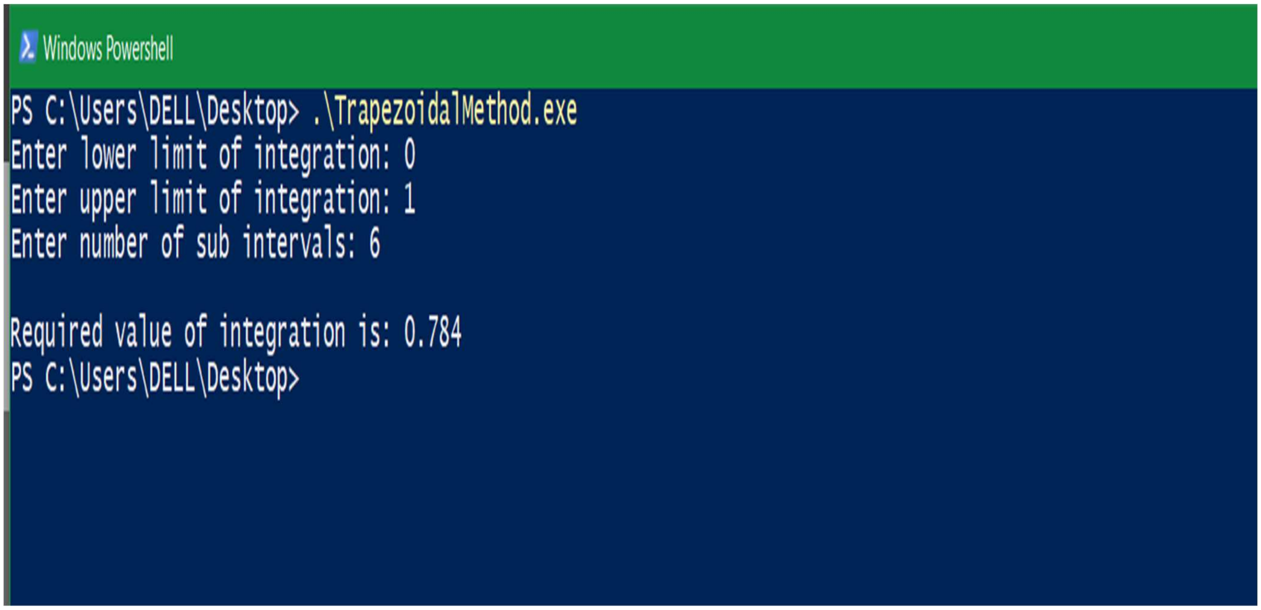
ALGORITHM-

```
1. Start
2. Define function f(x)
3. Read lower limit of integration, upper limit of
   integration and number of sub interval
4. Calcultae: step size = (upper limit - lower limit)/number of sub interval
5. Set: integration value = f(lower limit) + f(upper limit)
6. Set: i = 1
7. If i > number of sub interval then goto
8. Calculate: k = lower limit + i * h
9. Calculate: Integration value = Integration Value + 2* f(k)
10. Increment i by 1 i.e. i = i+1 and go to step 7
11. Calculate: Integration value = Integration value * step size/2
12. Display Integration value as required answer
13. Stop
```

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x) 1/(1+pow(x,2))

int main(){
 float lower, upper, integration=0.0, stepSize, k;
 int i, subInterval;
 clrscr();
 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;
  integration = integration + 2 * f(k);
 }
 integration = integration * stepSize/2;
 printf("\nRequired value of integration is: %.3f", integration);
 getch();
 return 0;
}
```

Output



```
PS C:\Users\DELL\Desktop> .\TrapezoidalMethod.exe
Enter lower limit of integration: 0
Enter upper limit of integration: 1
Enter number of sub intervals: 6

Required value of integration is: 0.784
PS C:\Users\DELL\Desktop>
```

Program 15

Objective: Program to Implement Simpson’s 1/3 Rule

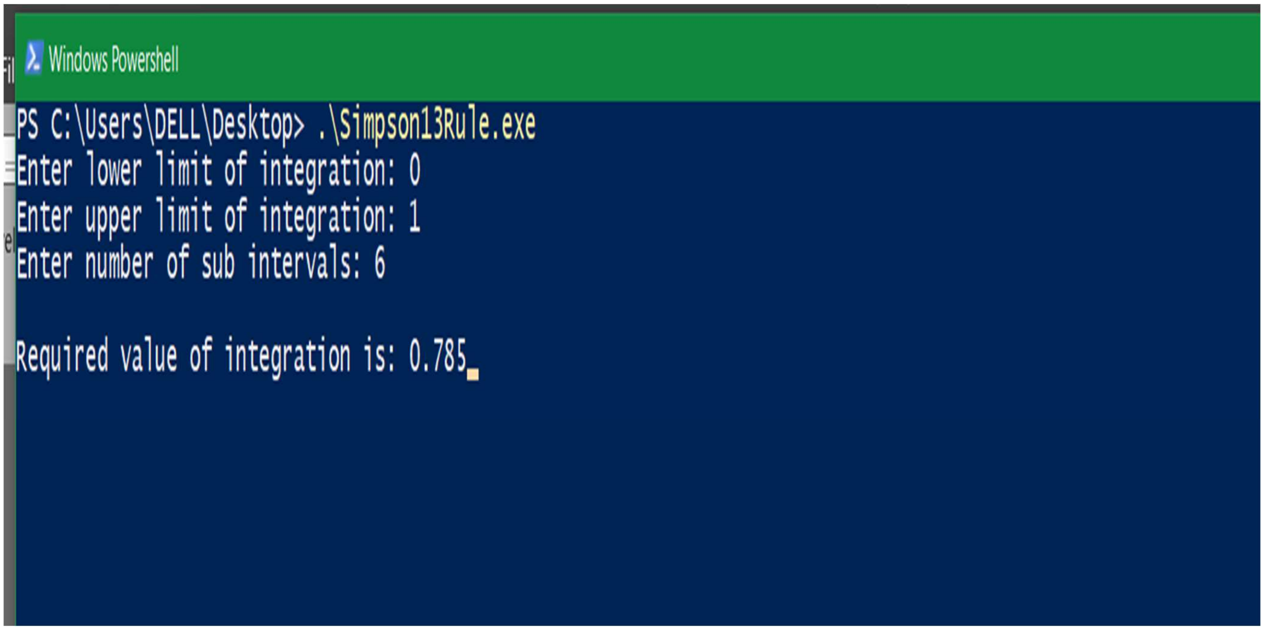
ALGORITHM-

```
1. Start
2. Define function f(x)
3. Read lower limit of integration, upper limit of
   integration and number of sub interval
4. Calcultae: step size = (upper limit - lower limit)/number of sub interval
5. Set: integration value = f(lower limit) + f(upper limit)
6. Set: i = 1
7. If i > number of sub interval then goto
8. Calculate: k = lower limit + i * h
9. If i mod 2 =0 then
     Integration value = Integration Value + 2* f(k)
   Otherwise
     Integration Value = Integration Value + 4 * f(k)
   End If
10. Increment i by 1 i.e. i = i+1 and go to step 7
11. Calculate: Integration value = Integration value * step size/3
12. Display Integration value as required answer
13. Stop
```

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;
 clrscr();
 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%2==0)
  {
   integration = integration + 2 * f(k);
  }
  else
  {
   integration = integration + 4 * f(k);
  }
 }
 integration = integration * stepSize/3;
 printf("\nRequired value of integration is: %.3f", integration);
 getch();
 return 0;
}
```

Output



```
PS C:\Users\DELL\Desktop> .\Simpson13Rule.exe
Enter lower limit of integration: 0
Enter upper limit of integration: 1
Enter number of sub intervals: 6

Required value of integration is: 0.785
```

Program 16

Objective: Program to Implement Simpson's 3/8 Rule

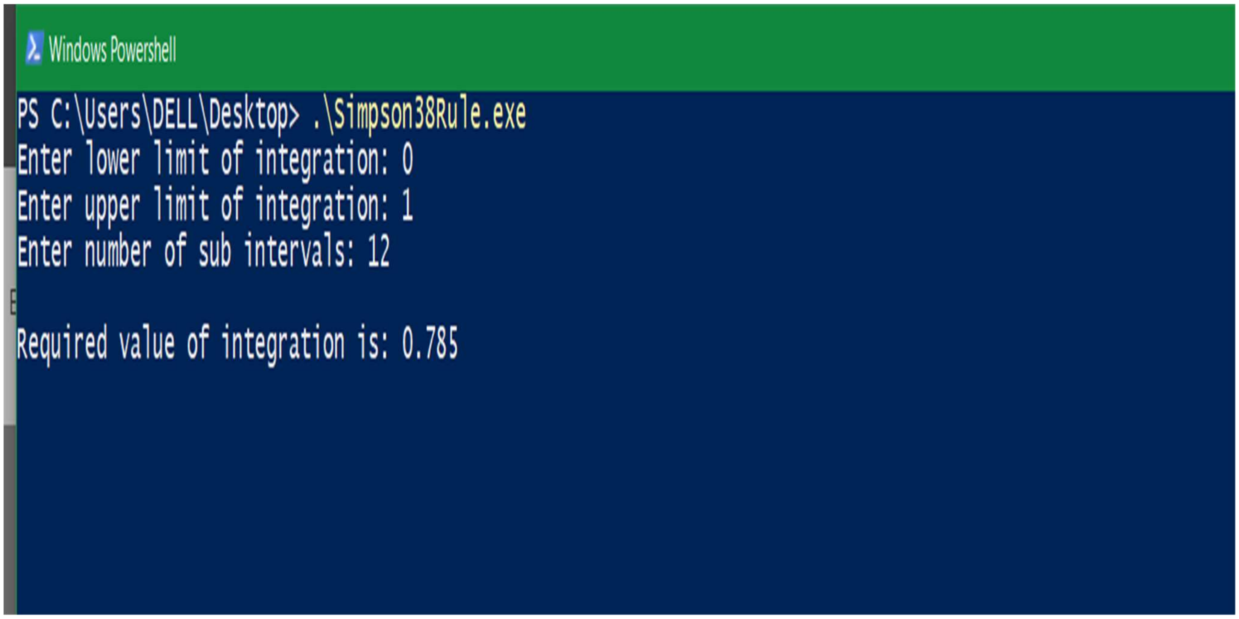
ALGORITHM-

```
1. Start
2. Define function f(x)
3. Read lower limit of integration, upper limit of
   integration and number of sub interval
4. Calcultae: step size = (upper limit - lower limit)/number of sub interval
5. Set: integration value = f(lower limit) + f(upper limit)
6. Set: i = 1
7. If i > number of sub interval then goto
8. Calculate: k = lower limit + i * h
9. If i mod 3 =0 then
     Integration value = Integration Value + 2* f(k)
   Otherwise
     Integration Value = Integration Value + 3 * f(k)
   End If
10. Increment i by 1 i.e. i = i+1 and go to step 7
11. Calculate: Integration value = Integration value * step size*3/8
12. Display Integration value as required answer
13. Stop
```

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;
 clrscr();

 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



```
PS C:\Users\DELL\Desktop> .\Simpson38Rule.exe
Enter lower limit of integration: 0
Enter upper limit of integration: 1
Enter number of sub intervals: 12

Required value of integration is: 0.785
```

Program 17

Objective: Program to Fit a Line

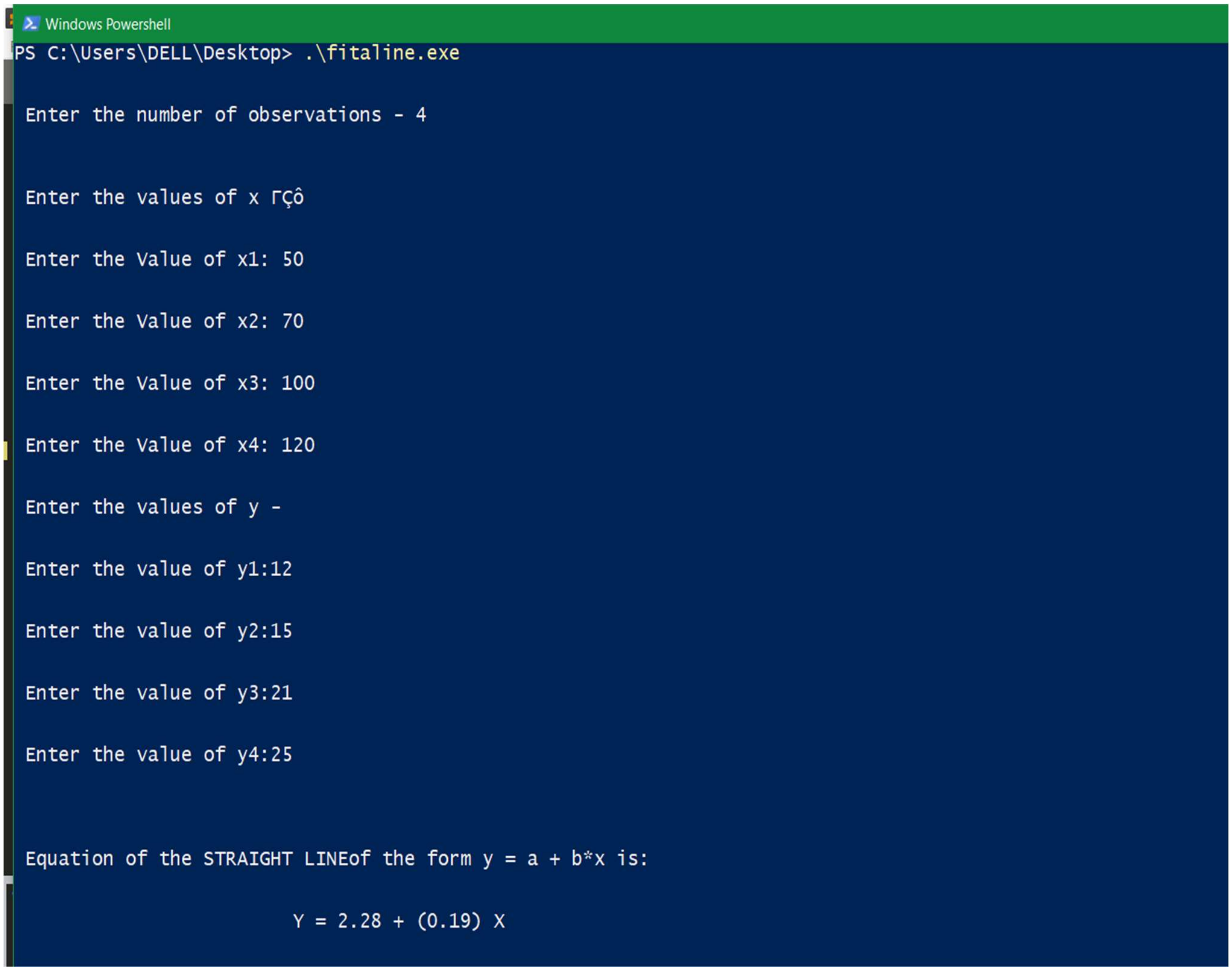
ALGORITHM-

```
1. Start
2. Input no. of terms observ
3. Input the array ax
4. Input the array ay
5. for i=0 to observ
6. sum1+=x[i]
7. sum2+=y[i]
8. xy[i]=x[i]*y[i]
9. sum3+=xy[i]
10. End Loop i
11. for i = 0 to observ
12. x2[i]=x[i]*x[i]
13. sum4+=x2[i]
14. End of Loop i
15. temp1=(sum2*sum4)-(sum3*sum1)
16. a=temp1/((observ *sum4)-(sum1*sum1))
17. b=(sum2-observ*a)/sum1
18. Print output a,b
19. Print “line is: y = a+bx”
20. Stop
```

Source Code

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
int main()
{
 int i=0;
 int observ;
 float x[10];
 float y[10];
 float xy[10];
 float x2[10];
 float sum1=0.0;
 float sum2=0.0;
 float sum3=0.0;
 float sum4=0.0;
 double a;
 double b;
 clrscr ();
 printf("\n\n Enter the number of observations - ");
 scanf("%d" ,&observ);
 printf("\n\n\n Enter the values of x – \n");
 for (i=0;i<observ;i++)
 {
 printf("\n\n Enter the Value of x%d: ",i+1);
 scanf("%f" ,&x[i]);
 sum1 +=x[i];
 }
 printf("\n\n Enter the values of y - \n");
 for(i=0;i<observ;i++)
 {
 printf("\n\n Enter the value of y%d:",i+1);
 scanf("%f",&y[i]);
 sum2+=y[i];
 }
 for(i=0;i<observ;i++)
 {
 xy[i]=x[i]*y[i];
 sum3 +=xy[i];
 }
 for(i=0;i<observ; i++)
 {
 x2[i]=x[i]*x[i];
 sum4 += x2[i];
 }
 a=(sum2*sum4-sum3*sum1)/(observ*sum4-sum1*sum1);
 b=(sum2-observ*a)/sum1;
 printf("\n\n\n\n Equation of the STRAIGHT LINE");
 printf("of the form y = a + b*x is:");
 printf("\n\n\n \t\t\t Y = %.2f + (%.2f) X", a,b);
 printf("\n\n\n Press Enter to Exit");
 getch();
}
```

Output



```
PS C:\Users\DELL\Desktop> .\fitaline.exe

 Enter the number of observations - 4


 Enter the values of x ΓÇô

 Enter the Value of x1: 50

 Enter the Value of x2: 70

 Enter the Value of x3: 100

 Enter the Value of x4: 120

 Enter the values of y -

 Enter the value of y1:12

 Enter the value of y2:15

 Enter the value of y3:21

 Enter the value of y4:25


 Equation of the STRAIGHT LINEof the form y = a + b*x is:

                         Y = 2.28 + (0.19) X
```

Program 18

Objective: Program to Fit a Parabola

ALGORITHM-

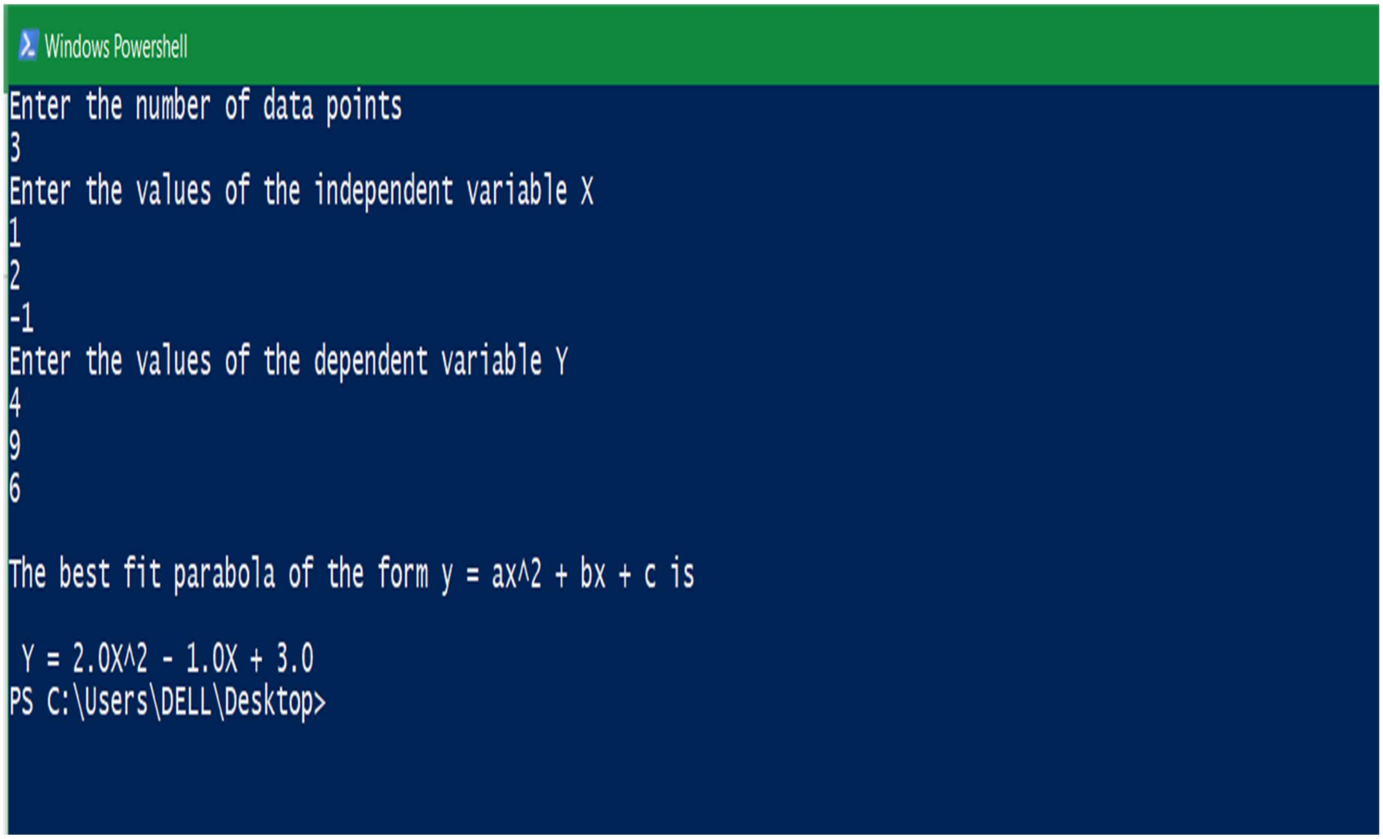
```
1. Start
2. Input no. of terms observ
3. Input the array ax
4. Input the array ay
5. for i=0 to observ
6. sum1+=x[i]
7. sum2+=y[i]
8. xy[i]=x[i]*y[i]
9. sum3+=xy[i]
10. End Loop i
11. for i = 0 to observ
12. x2[i]=x[i]*x[i]
13. sum4+=x2[i]
14. End of Loop i
15. t = dx / det
16. u = dy / det
17. v = dz / det
18. put t,u,v
19. Print “line is: y = ax^2 + bx + c”
20. Stop
```

Source Code

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

int n;
float x[100], y[100], XS = 0, YS = 0, XYS = 0, XSS = 0, XCS = 0, XQS = 0,
XSYS = 0, Xavg = 0, Yavg = 0;

int calculate()
{
    int i;
    for (i = 0; i < n; i++)
    {
        XS += x[i];
        YS += y[i];
        XYS += x[i] * y[i];
        XSS += x[i] * x[i];
        XCS += x[i] * x[i] * x[i];
        XQS += x[i] * x[i] * x[i] * x[i];
        XSYS += x[i] * x[i] * y[i];
    }

    Xavg = XS / n;
    Yavg = YS / n;
    return 0;
}

int main()
{
    int i, choice;
    float NR, DR, p, q, r;
    float m, c;
    float t, u, v, det, dx, dy, dz;

    printf("Enter the number of data points\n");
    scanf("%d", &n);
    printf("Enter the values of the independent variable X\n");
    for (i = 0; i < n; i++)
    {
        scanf("%f", &x[i]);
    }
    printf("Enter the values of the dependent variable Y\n");
    for (i = 0; i < n; i++)
    {
        scanf("%f", &y[i]);
    }

    calculate();
```

```
        det =  XSS * (XSS * XSS - XCS * XS) - XS * (XCS * XSS - XS * XQS)
+ n * (XCS * XCS - XQS * XSS);
        dx  =  YS * (XSS * XSS - XS * XCS) - XS * (XYS * XSS - XS * XSYS) +
n * (XYS * XCS - XSS * XSYS);
        dy  =  XSS * (XYS * XSS - XS * XSYS) - YS * (XCS * XSS - XS * XQS)
+ n * (XCS * XSYS - XQS * XYS);
        dz  =  XSS * (XSS * XSYS - XYS * XCS) - XS * (XCS * XSYS - XYS *
XQS) + YS * (XCS * XCS - XQS * XSS);

        t = dx / det;
        u = dy / det;
        v = dz / det;

        if (u >= 0 && v >= 0)
           printf("\nThe best fit parabola of the form y = ax^2 + bx + c is\n\n Y =
%.1fX^2 + %.1fX + %.1f\n", t, u, v);
        else if (u < 0 && v > 0)
           printf("\nThe best fit parabola of the form y = ax^2 + bx + c is\n\n Y =
%.1fX^2 - %.1fX + %.1f\n", t, -1 * u, v);
        else if (u > 0 && v < 0)
        {
           printf("\nThe best fit parabola of the form y = ax^2 + bx + c is\n\n Y =
%.1fX^2 + %.1fX - %.1f\n", t, u, -1 * v);
        }
        else
        {
           printf("\nThe best fit parabola of the form y = ax^2 + bx + c is\n\n Y =
%.1fX^2 - %.1fX - %.1f\n", t, -1 * u, -1 * v);
        }
return 0;
}
```

Output



```
Enter the number of data points
3
Enter the values of the independent variable X
1
2
-1
Enter the values of the dependent variable Y
4
9
6

The best fit parabola of the form y = ax^2 + bx + c is

 Y = 2.0X^2 - 1.0X + 3.0
PS C:\Users\DELL\Desktop>
```