NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find the real root of the equation $x^3 - 2x - 5 = 0$ by using Newton Raphson Method.

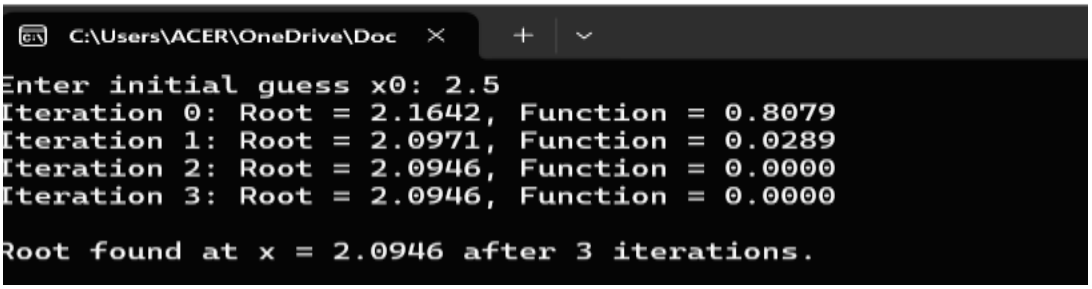
OBJECTIVE: To implement the concept of Newton Raphson Method to find the real root of the given equation.

CODE:-

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

#define f(x) (x*x*x - 2*x - 5)
#define df(x) (3*x*x - 2)
#define e 0.0001
void main() {
   int iterations = 0;
   float x0, x1, f0, f1;

   printf("Enter initial guess x0: ");
   scanf("%f", &x0);

   do {
      f0 = f(x0);
      f1 = df(x0);

      x1 = x0 - (f0 / f1);

      printf("Iteration %d: Root = %.4f, Function = %.4f\n", iterations, x1, f(x1));

      if (fabs(x1 - x0) < e) {
         break;
      }
x0 = x1;
      iterations++;
```

```
    } while (1);

     printf("\nRoot found at x = %.4f after %d iterations.\n", x1, iterations);

}
```

OUTPUT:-



NAME: ANUJ NEGI COURSE: BCA 'D1'
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find the real root of the equation $x^3 - 2x - 5 = 0$ by using Bisection Method.

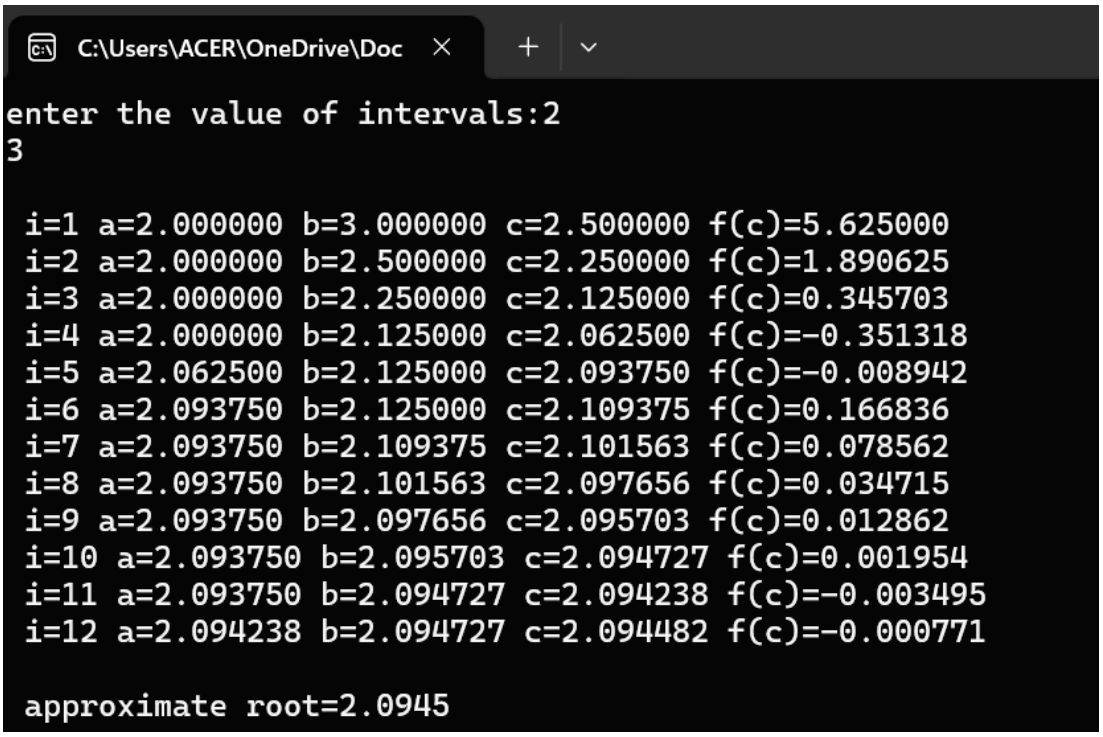
OBJECTIVE: To implement the concept of Bisection Method to find the real root of the given equation.

CODE:-

```
#include<stdio.h>
#include<math.h>
#define f(x) (x*x*x-2*x-5)
#define e 0.001
void main()
        {
          int  i=1;
          float a,b,c,f;
          printf("enter the value of intervals:");
          scanf("%f %f",&a,&b);
                do{
                    c=(a+b)/2;
           f=f(c);
           printf("\n i=%d a=%f b=%f c=%f f(c)=%f",i,a,b,c,f);
           if(f(c)<0){
               a=c;
             }
                    else{
            b=c;
           }
           i++;
         }
   while (fabs(f(c))>e);
   printf("\n \n approximate root=%.4f\n",c);
  }
```

NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

OUTPUT:-



NAME: ANUJ NEGI COURSE: BCA 'D1'
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find the real root of the given equation $x^3 - 2x - 5 = 0$ by using Regula – Falsi (False position) Method.

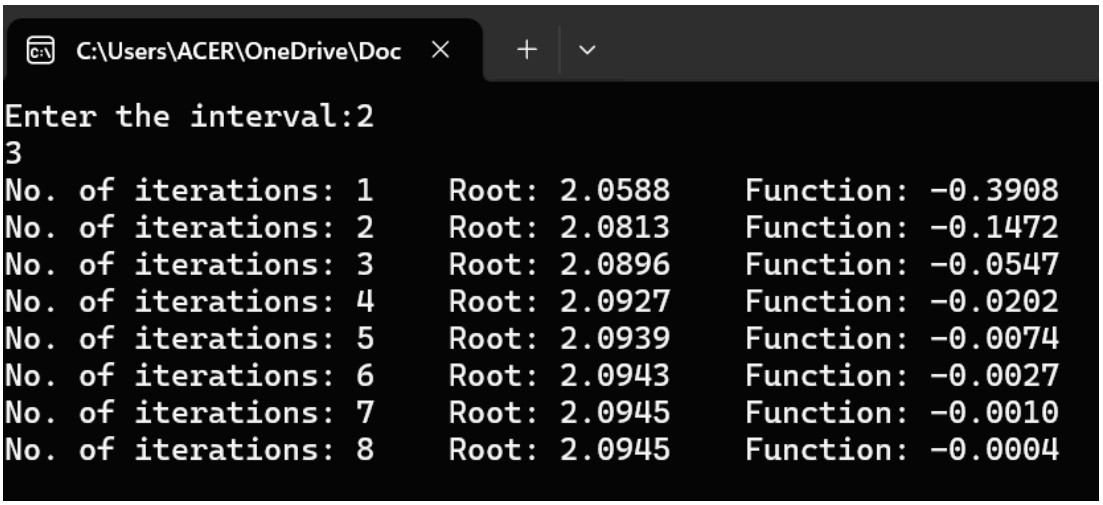
OBJECTIVE: To implement the concept of Regula-Falsi (False Position) Method to find the real root of the given equation.

CODE:-

```
#include<stdio.h>
#include<math.h>
#define f(x) (x*x*x-2*x-5)
#define e 0.001
void main()
{
   int i=0;
   float x0,x1,x2,f0,f1,f2;
   printf("Enter the interval:");
   scanf("%f %f",&x0,&x1);
   do {
      f0 = f(x0);
      f1 = f(x1);
      x2 = x1 - ((f1*(x1-x0))/(f1-f0));
      f2 = f(x2);
      if(f0*f2<0) {
         x1 = x2;
         f1 = f2;
      } else {
         x0 = x2;
         f0 = f2;
      }
      i++;
      printf("No. of iterations: %d\t", i);
      printf("Root: %.4f\t", x2);
      printf("Function: %.4f\n", f2);
```

```
    } while(fabs(f2) > e);


}
```

OUTPUT:-



NAME: ANUJ NEGI COURSE: BCA 'D1'
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

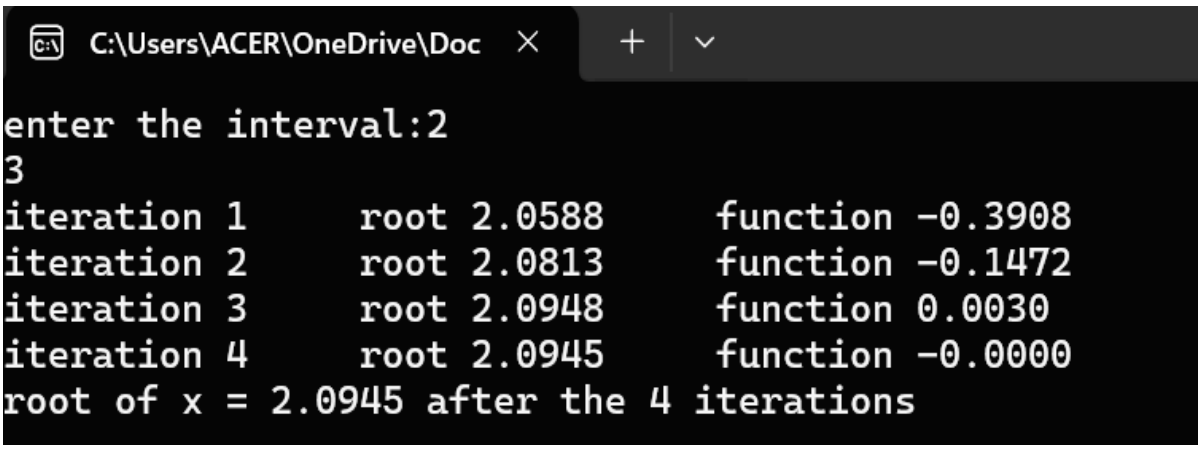
PROBLEM STATEMENT: Write a c program to find the real root of the equation $x^3 - 2x - 5 = 0$ by using Secant Method.

OBJECTIVE: To implement the concept of Secant Method to find the real root of the given equation.

CODE:-

```
#include<stdio.h>
#include<math.h>
#define f(x) (x*x*x-2*x-5)
#define e 0.0001
void main()
{
int i=0;
float x0,x1,x2,f0,f1,f2;
printf("enter the interval:");
scanf("%f %f",&x0,&x1);
do{
f0=f(x0);
f1=f(x1);
x2=(x1-(f1*(x0-x1))/(f0-f1));
f2=f(x2);
x0=x1;
f0=f1;
x1=x2;
f1=f2;
i++;
printf("iteration %d\t",i);
printf("root %.4f\t",x2);
printf("function %.4f\n",f2);
}
while(fabs(f2)>0.0001);
printf("root of x = %.4f after the %d iterations",x2,i); }
```

OUTPUT:-


C:\Users\ACER\OneDrive\Doc
enter the interval:2
3
iteration 1 root 2.0588 function -0.3908
iteration 2 root 2.0813 function -0.1472
iteration 3 root 2.0948 function 0.0030
iteration 4 root 2.0945 function -0.0000
root of x = 2.0945 after the 4 iterations

NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to compute error, absolute error, relative error and relative error percentage.

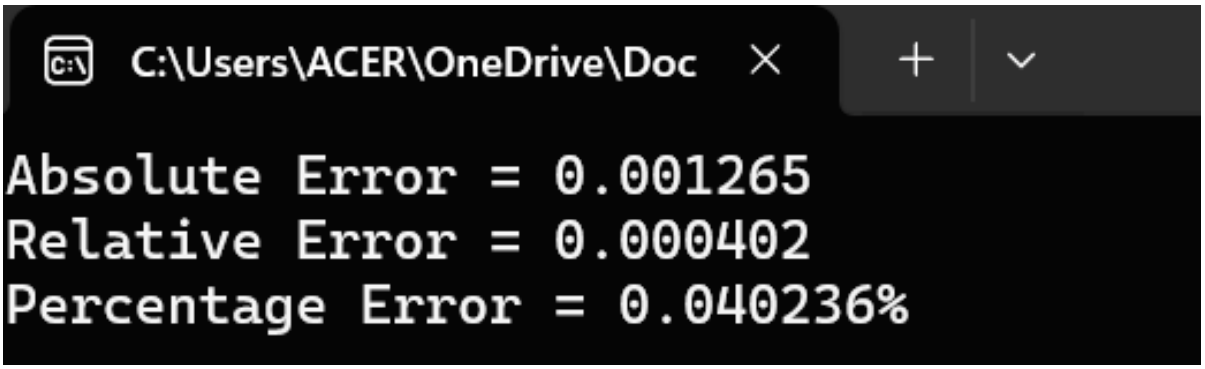
(take true value of pi = 3.1415926 and approx. value of pi = 22.0/7)

OBJECTIVE: To implement the concept of Absolute Error , Relative Error , Percentage Error.

CODE:-

```
#include <stdio.h>
#include <math.h>
void main() {
    float pi = 22.0 / 7;
    float approx_pi = 3.1415926;
    float Ea = fabs(pi - approx_pi);
    float Er = Ea / pi;
    float Ep = Er * 100;
 printf("Absolute Error = %f\nRelative Error = %f\nPercentage Error = %f%%", Ea, Er, Ep);
}
```

OUTPUT:-



NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find the real root of the equation $x^3 - 2x - 5 = 0$ by using Iteration Method (Fixed-Point).

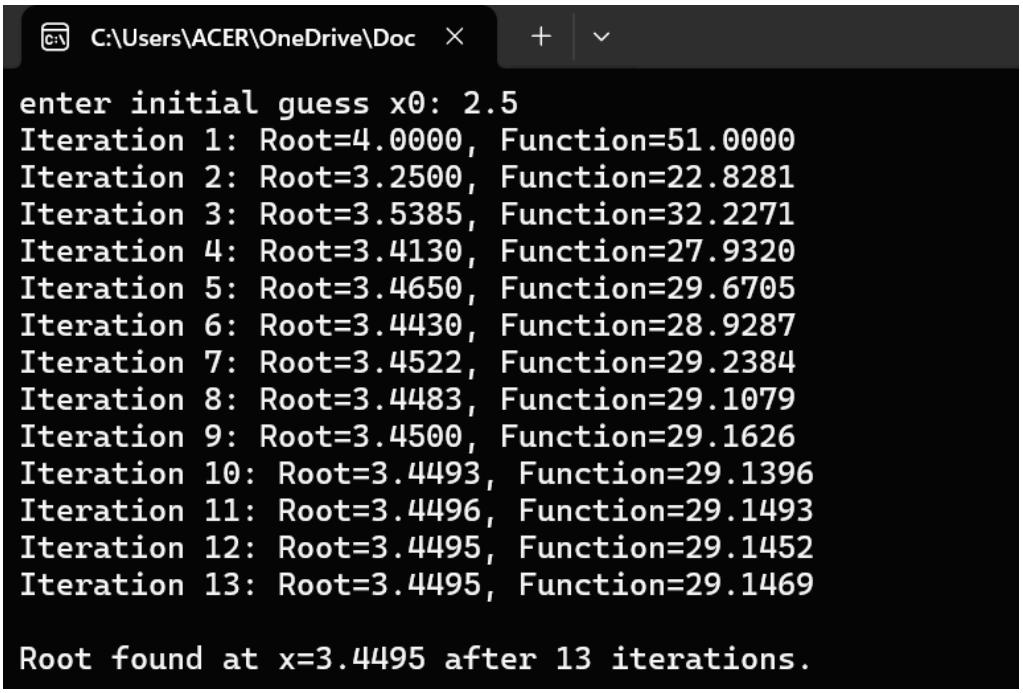
OBJECTIVE: To implement the concept of Iteration Method (Fixed-Point) to find the real root of the given equation.

CODE:-

```
#include<stdio.h>
#include<math.h>
#define f(x) (x*x*x-2*x-5)
#define g(x) (2*x+5)/x
#define e 0.0001
void main()
{
   int i=1;
   float x0,x1;
   printf("enter initial guess x0: ");
   scanf("%f",&x0);
   do
   {
      x1=g(x0);
      printf("Iteration %d: Root=%.4f, Function=%.4f\n",i,x1,f(x1));
      if(fabs(x1-x0)<e)
      {
         break;
      }
      x0=x1;
      i++;
   }
   while(1);
   printf("\nRoot found at x=%.4f after %d iterations.\n",x1,i);
}
```

NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

OUTPUT:-



NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find approximate value of a definite integral using Trapezoidal rule.

OBJECTIVE: To implement the concept of Trapezoidal rule to find the approximate value of definite integral

```
#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;

 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);

 return 0;}
```

NAME: ANUJ NEGI
ROLL.NO: 10

COURSE: BCA ‘D1’
SUBJECT: CBNST (PBC 402)

OUTPUT:

```
/tmp/vLFIQ7rZOB.o
Enter lower limit of integration: 0
Enter upper limit of integration: 6
Enter number of sub intervals: 6

Required value of integration is: 1.411
```

NAME: ANUJ NEGI COURSE: BCA 'D1'
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find approximate value of a definite integral using Simpson's 1/3 rule.

OBJECTIVE: To implement the concept of Simpson's 1/3 rule to find the approximate value of definite integral

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,h, 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);

h = (upper - lower)/subInterval;

integration = f(lower) + f(upper);

 for(i=1; i<= subInterval-1; i++)
 {
  k = lower + i*h;
  if(i%2==0)
  {
   integration = integration + 2 * f(k);
  }
  else
  {
   integration = integration + 4 * f(k);
  }
 }
 integration = integration * h/3;

 printf("\nRequired value of integration is: %.3f", integration);
 return 0;
```

}

OUTPUT:

```
Enter lower limit of integration: 0
Enter upper limit of integration: 6
Enter number of sub intervals: 6

Required value of integration is: 1.366
```

NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find approximate value of a definite integral using Simpson’s 3/8 rule.

OBJECTIVE: To implement the concept of Simpson’s 3/8 rule to find the approximate value of definite integral

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f(x) 1/(1+x*x)

int main()
{
 float lower, upper, integration=0.0,h, 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);

 h = (upper - lower)/subInterval;

 integration = f(lower) + f(upper);

 for(i=1; i<= subInterval-1; i++)
 {
  k = lower + i*h;
  if(i%3 == 0)
  {
   integration = integration + 2 * f(k);
  }
  else
  {
   integration = integration + 3 * f(k);
  }
 }
 integration = integration * h*3/8;

 printf("\nRequired value of integration is: %.3f", integration);
```

```
return 0;
}
```

OUTPUT:

```
Enter lower limit of integration: 1
Enter upper limit of integration: 6
Enter number of sub intervals: 6

Required value of integration is: 0.624
```

NAME: ANUJ NEGI COURSE: BCA ‘D1’
ROLL.NO: 10 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to find real root of the equation using Newton’s Raphson method.

OBJECTIVE: To implement the concept of Newton’s Raphson method to find the find real root of the equation.

```
#include <stdio.h>

void forward(float x[4], float y[4][4], int n);

int main()
{
	int i, j;

	int n = 4;

	float x[4] = { 0, 1, 2, 3 };

	float y[4][4] = {

		{ 1, 0, 0, 0 },
		{ 0, 0, 0, 0 },
		{ 1, 0, 0, 0 },
		{ 10, 0, 0, 0 },

};
	forward(x, y, n);

	return 0;
}
void forward(float x[4], float y[4][4], int n)
{
	int i, j;

	float a = 0.5; // interpolation point

	float h, u, sum, p;

	for (j = 1; j < n; j++) {

		for (i = 0; i < n - j; i++) {

			y[i][j] = y[i + 1][j - 1] - y[i][j - 1];
		}
	}
	printf("\n The forward difference table is:\n");

	for (i = 0; i < n; i++) {

		printf("\n");

		for (j = 0; j < n - i; j++) {
```

```
                printf("%f\t", y[i][j]);
            }
        }

        p = 1.0;

        sum = y[0][0];

        h = x[1] - x[0];

        u = (a - x[0]) / h;

        for (j = 1; j < n; j++)
{
            p = p * (u - j + 1) / j;
            sum = sum + p * y[0][j];
        }
        printf("\nThe value of y at x=%0.1f is %0.3f", a, sum);
}
```

OUTPUT:

```
 The forward difference table is:

1.000000    -1.000000   2.000000    6.000000
0.000000    1.000000    8.000000
1.000000    9.000000
10.000000
The value of y at x=0.5 is 0.625
```

NAME: ASHISH KOTHARI COURSE: BCA 'D1'
ROLL.NO: 17 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to interpolate the value of x by using newtons backward formula.

OBJECTIVE: To implement the concept of Newtons backward formula.

CODE:

```
void BackWard(float x[4], float y[4][4], int n);
int main()
{
    int i, j;
    int n = 4; // number of arguments
    float x[4] = { 0, 1, 2, 3 };
    float y[4][4] = {
        { 1, 0, 0, 0 },
        { 0, 0, 0, 0 },
        { 1, 0, 0, 0 },
        { 10, 0, 0, 0 },
    };
    BackWard(x, y, n);

    return 0;
}
void BackWard(float x[4], float y[4][4], int n)
{
    int i, j;
    float a = 0.5; // interpolation point
    float h, u, sum, p;
    for (j = 1; j < n; j++) {
        for (i = j; i < n; i++) {
            y[i][j] = y[i][j - 1] - y[i - 1][j - 1];
        }
    }
    printf("\nThe backward difference table is:\n");
    for (i = 0; i < n; i++) {
        printf("\n");
        for (j = 0; j <= i; j++) {
            printf("%f\t", y[i][j]);
        }
    }

    p = 1.0;
    sum = y[n - 1][0];
    h = x[1] - x[0];
    u = (a - x[n - 1]) / h;
    for (j = 1; j < n; j++) {
        p = p * (u + j - 1) / j;
        sum = sum + p * y[n - 1][j];
    }

    printf("\nThe value of y at x=%0.1f is %0.3f", a, sum);
}
```

Output:-

```
The backward difference table is:

1.000000
0.000000    -1.000000
1.000000    1.000000    2.000000
10.000000    9.000000    8.000000    6.000000
The value of y at x=0.5 is 0.625
```

NAME: ASHISH KOTHARI COURSE: BCA 'D1'
ROLL.NO: 17 SUBJECT: CBNST (PBC 402)

PROBLEM STATEMENT: Write a c program to interpolate the value of x by using newtons backward formula.

OBJECTIVE: To implement the concept of Newtons backward formula.

CODE:

```
void BackWard(float x[4], float y[4][4], int n);
int main()
{
    int i, j;
    int n = 4; // number of arguments
    float x[4] = { 0, 1, 2, 3 };
    float y[4][4] = {
        { 1, 0, 0, 0 },
        { 0, 0, 0, 0 },
        { 1, 0, 0, 0 },
        { 10, 0, 0, 0 },
    };
    BackWard(x, y, n);

    return 0;
}
void BackWard(float x[4], float y[4][4], int n)
{
    int i, j;
    float a = 0.5; // interpolation point
    float h, u, sum, p;
    for (j = 1; j < n; j++) {
        for (i = j; i < n; i++) {
            y[i][j] = y[i][j - 1] - y[i - 1][j - 1];
        }
    }
    printf("\nThe backward difference table is:\n");
    for (i = 0; i < n; i++) {
        printf("\n");
        for (j = 0; j <= i; j++) {
            printf("%f\t", y[i][j]);
        }
    }

    p = 1.0;
    sum = y[n - 1][0];
    h = x[1] - x[0];
    u = (a - x[n - 1]) / h;
    for (j = 1; j < n; j++) {
        p = p * (u + j - 1) / j;
        sum = sum + p * y[n - 1][j];
    }

    printf("\nThe value of y at x=%0.1f is %0.3f", a, sum);
}
```

Output:-

```
The backward difference table is:

1.000000
0.000000    -1.000000
1.000000    1.000000    2.000000
10.000000    9.000000    8.000000    6.000000
The value of y at x=0.5 is 0.625
```

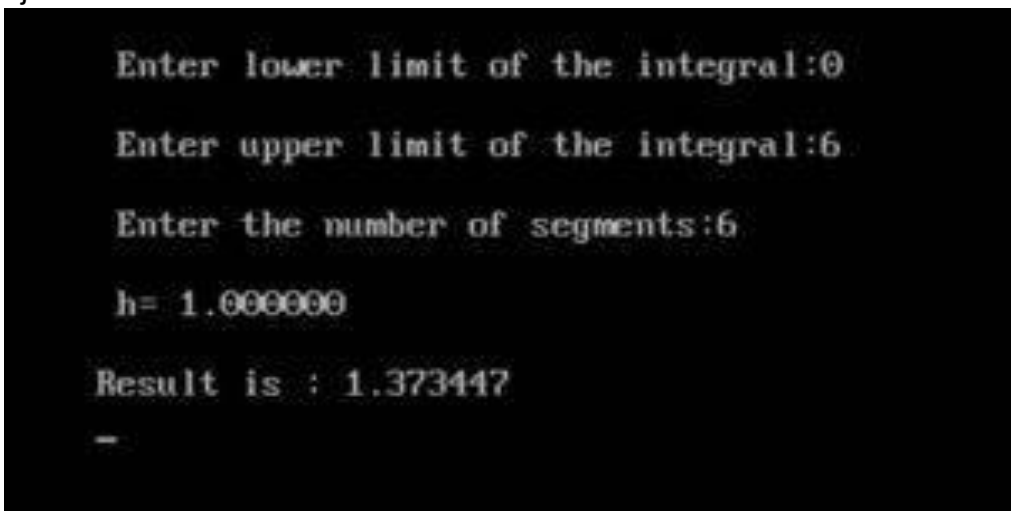
Name- Ashish Kothari Subject- CBNST Lab
Section- D1 Subject Code-PBC 402
Roll no- 17 University Rollno-2221283

Weddle's Rule:

Objective: Write a c program to implement the given numerical integration equation using Widdle's rule.

```
#include<stdio.h>
#include<math.h>
float f(float x)
{
   float y;
   y=1/(1+x*x);
   return(y);
}
 int main()
{
   float a,b,h,s1=0,s2=0,s=0;
   int i,n,m;
   printf("Enter the value of upper limit= ");
   scanf("%f",&b);
   printf("Enter the value of lower limit= ");
   scanf("%f",&a);
   printf("Enter the value of n=");
   scanf("%d",&n);
   h=(b-a)/n;
   printf("h= %f",h);
   m=n/6;
   s=0;
   if(n%6==0)
```

```
  {
     for(i=1; i<=m; i++)
     {
s=s+((3*h/10)*(f(a)+f(a+2*h)+5*f(a+h)+6*f(a+3*h)+f(a+4*h)+5*f(a+5*h)+f(a+6
*h)));
        a=a+6*h;
     }
     printf("Result is : %f",s);
  }
  else
  {
     printf(" Weddle's rule is not applicable");
  }
  return 0;
}
```



Name- Ashish Kothari
Section- D1
Roll no- 17

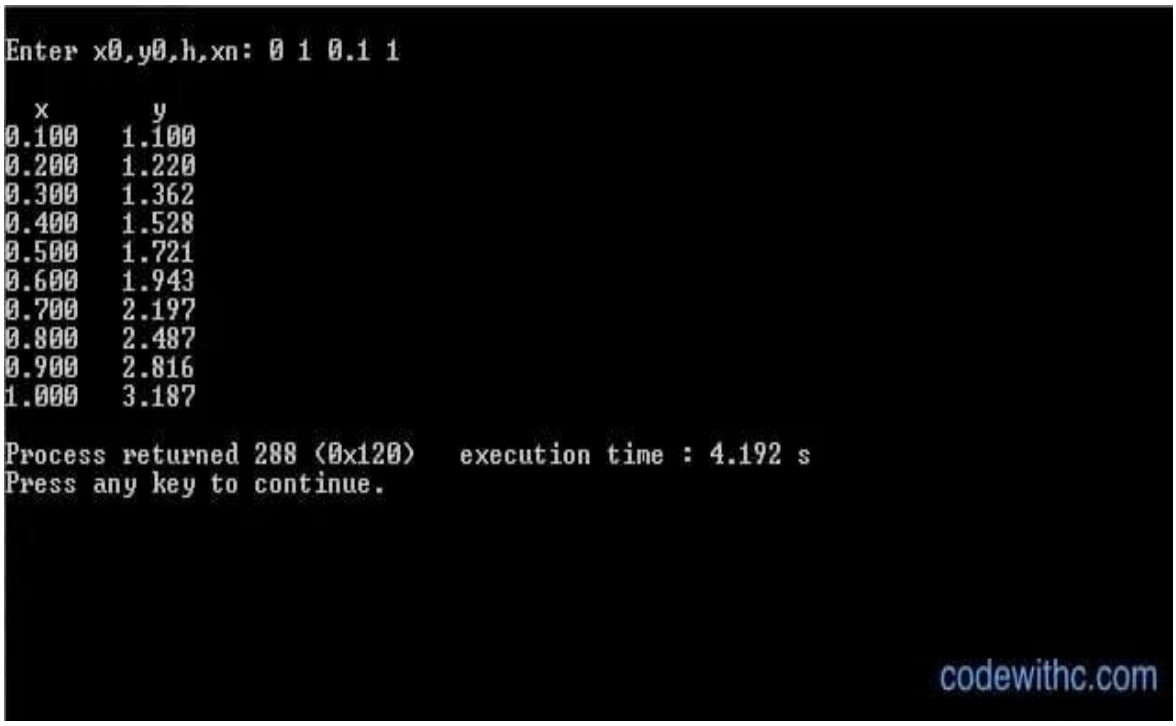
Subject- CBNST Lab
Subject Code-PBC 402
University Rollno-2221283

Euler's Method:

Objective: Write a c program to implement the given ordinary equation using Euler's method.

```
#include<stdio.h>
float fun(float x,float y)
{
   float f;
   f=x+y;
   return f;
}
int main()
{
   float a,b,x,y,h,t,k;
   printf("\nEnter x0,y0,h,xn: ");
   scanf("%f%f%f%f",&a,&b,&h,&t);
   x=a;
   y=b;
   printf("\n x\t y\n");
   while(x<=t)
   {
      k=h*fun(x,y);
      y=y+k;
      x=x+h;
      printf("%0.3f\t%0.3f\n",x,y);
   }
```

```
    return 0;
}
```


Enter x0,y0,h,xn: 0 1 0.1 1
x y
0.100 1.100
0.200 1.220
0.300 1.362
0.400 1.528
0.500 1.721
0.600 1.943
0.700 2.197
0.800 2.487
0.900 2.816
1.000 3.187
Process returned 288 (0x120) execution time : 4.192 s
Press any key to continue.
codewithc.com

Name- Ashish Kothari
Section- D1
Roll no- 17
Subject- CBNST Lab
Subject Code-PBC 402
University Rollno-2221283

Runge Kutta Method of 4th order:

Objective: Write a c program to implement the given ordinary equation using Runge Kutta Method of 4th order.

```
#include<stdio.h>
#include<math.h>
#define f(x,y) (y*y-x*x)/(y*y+x*x)
int main()
{
 float x0, y0, xn, h, yn, k1, k2, k3, k4, k;
 int i, n;
 printf("Enter Initial Condition\n");
 printf("x0 = ");
 scanf("%f", &x0);
 printf("y0 = ");
 scanf("%f", &y0);
 printf("Enter calculation point xn = ");
 scanf("%f", &xn);
 printf("Enter number of steps: ");
 scanf("%d", &n);
 /* Calculating step size (h) */
 h = (xn-x0)/n;
 /* Runge Kutta Method */
 printf("\nx0\ty0\tyn\n");
 for(i=0; i < n; i++)
 {
```

```
  k1 = h * (f(x0, y0));
  k2 = h * (f((x0+h/2), (y0+k1/2)));
  k3 = h * (f((x0+h/2), (y0+k2/2)));
  k4 = h * (f((x0+h), (y0+k3)));
  k = (k1+2*k2+2*k3+k4)/6;
  yn = y0 + k;
  printf("%0.4f\t%0.4f\t%0.4f\n",x0,y0,yn);
  x0 = x0+h;
  y0 = yn;
 }
 /* Displaying result */
 printf("\nValue of y at x = %0.2f is %0.3f",xn, yn);
 return 0;
}
```

Name- Ashish Kothari
Section- D1
Roll no- 17

Subject- CBNST Lab
Subject Code-PBC 402
University Rollno-2221283