# PROGRAM NO. 1

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech (CSE)

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

## 1.OBJECTIVE

Write a program in C to calculate the absolute error, relative error and percentage error using truncation and round off concepts.

## 2.METHOD /ALGORITHM

1. Enter the value of x.
2. Calculate the truncated value and the rounded off value.
3. Calculate the Absolute Error $E_a$ = | X – X'|.
4. Calculate the Relative Error $E_r$ = | X – X'| / | X |.
5. Calculate the Percentage Error $E_p = E_r$ * 100.
6. Display $E_a$, $E_r$ , $E_p$ .

## 3.PROGRAM

```
#include <stdio.h>

#include <math.h>

int main()

{

   double x, temp, x1;

   int n;

   printf("Enter the number : ");
```

```
    scanf("%lf", &x);
    printf("Enter the decimal place  : ");
    scanf("%d", &n);
    temp = x * pow(10, n);
    int res = (int)(temp + 0.5);
    int res1 = (int)(temp);
    temp = (float)res / pow(10, n);
    x1 = (float)res1 / pow(10, n);
    printf("After round off we have : %lf\n", temp);
    printf("After truncate we have : %lf\n", x1);
    printf("absolute error %f \n", fabs(x - x1));
    printf("relative error %f \n", fabs((x - x1) / x));
    printf("percentage error %f \n", fabs((x - x1) / x) * 100);
}
```

**4.OUTPUT**

```
Enter the number : 4.543432343
Enter the decimal place  : 4
After round off we have : 4.543400
After truncate we have : 4.543400
absolute error 0.000032
relative error 0.000007
percentage error 0.000712
```

# PROGRAM NO.2

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech (CSE)

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

## 1. 1.OBJECTIVE

Write a program in C to find the root of any transcendental equation using bisection method correct upto 3 decimal places.

## 2.METHOD /ALGORITHM

1. Read x1, x2, e
   *Here x1 and x2 are initial guesses
   e is the absolute error i.e. the desired degree of accuracy*
2. Compute: f1 = f(x1) and f2 = f(x2)
3. If (f1*f2) > 0, then display initial guesses are wrong
4. Otherwise, continue.
5. x = (x1 + x2)/2
6. If ( [ (x1 – x2)/x ] < e ), then display x
   * 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.

**3.PROGRAM**

```
#include <stdio.h>
#include <math.h>
float fun(float x)
{
   return (pow(x, 3) - (4 * x) - 9);
}
int main()
{
   int m, n, x1, x2, x3;
   float a, b;

   printf("enter the range ");
   scanf("%d %d", &m, &n);
   for (int i = m; i + 1 < n; i++)
   {
      if ((fun((float)i)) * (fun((float)(i + 1))) < 0)
      {
         a = (float)i;
         b = (float)(i + 1);
         break;
      }
   }
   printf("a is %f and b is %f\n", a, b);
   int i = 1;
```

```
    float x = (a + b) / 2;
    do
    {
        printf("value at iteration x%d is %f\n ", i, x);
        if (fun(a) * fun(x) < 0)
            b = x;
        else if (fun(x) * fun(b) < 0)
            a = x;
        x = (a + b) / 2;
        i++;
    } while (fabs(x - a) >= 0.0001 || fabs(x - b) >= 0.0001);
}
```

**4.OUTPUT :**

enter the range 0 5

a is 2.000000 and b is 3.000000

value at iteration x1 is 2.500000

value at iteration x2 is 2.750000

value at iteration x3 is 2.625000

value at iteration x4 is 2.687500

value at iteration x5 is 2.718750

value at iteration x6 is 2.703125

value at iteration x7 is 2.710938

value at iteration x8 is 2.707031

value at iteration x9 is 2.705078

value at iteration x10 is 2.706055

value at iteration x11 is 2.706543

value at iteration x12 is 2.706299

value at iteration x13 is 2.706421

# PROGRAM NO.3

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech (CSE)

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

## 1.OBJECTIVE

Write a program in C to find the solution of any transcendental equation using Regula-Falsi method correct up to three decimal places.

## 2.METHOD /ALGORITHM

1. 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*
2. Computer function values f(x0) and f(x1)
3. 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.
4. Determine: x = [x0*f(x1) – x1*f(x0)] / (f(x1) – f(x0))
5. 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;
6. 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.*
7. Display the root as x

**3.PROGRAM**

```
#include <stdio.h>
#include <math.h>
float function_return(float x)
{
   float res;
   res = pow(x, 3) - (4 * x) - 9;
   return res;
}
int find_points()
{
   float res;
   float res1;
   for (int i = 0; i < 5; i++)
   {
      res = function_return((float)i);
      res1 = function_return((float)(i + 1));
      if (res1 < 0 && res > 0 || res1 > 0 && res < 0)
         return (float)i;
   }
   return 0;
}
float truncate(float f, int p)
{
   float temp = f * pow(10, p);
```

```
    int res = (int)(temp);
    temp = (float)res / pow(10, p);
    return temp;
}
int main()
{
    float root, temp = 0.0, diff, mid, count;
    float f_i, s_i;
    int c = 1;
    f_i = find_points();
    s_i = f_i + 1;
    do
    {
        root = function_return(mid);
        mid = (f_i * function_return(s_i) - s_i * function_return(f_i))/(function_return(s_i) -
function_return(f_i));
        if (function_return(f_i) * function_return(mid) < 0)
            s_i = mid;
        else if (function_return(mid) * function_return(s_i) < 0)
            f_i = mid;
        printf("%d root is : %f\n", c, mid);
        diff = mid - temp;
        temp = mid;
        if (fabs(diff * 1000) < 1)
        {
```

```
            printf("root is : %f\n", truncate(mid, 3));
            break;
        }
        c++;
    } while (1);
    return 0;
}
```

**4.OUTPUT :**

```
enter the range 0 5
a is 2.000000 and b is 3.000000
value at iteration x1 is 2.500000
 value at iteration x2 is 2.750000
 value at iteration x3 is 2.625000
 value at iteration x4 is 2.687500
 value at iteration x5 is 2.718750
 value at iteration x6 is 2.703125
 value at iteration x7 is 2.710938
 value at iteration x8 is 2.707031
 value at iteration x9 is 2.705078
 value at iteration x10 is 2.706055
 value at iteration x11 is 2.706543
value at iteration x12 is 2.706299
 value at iteration x13 is 2.70642
```

# PROGRAM NO. 4

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech (CSE)

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

**1.OBJECTIVE:**

Write a program to find solution of any non polynomial equation using Newton Rapshon Method.

**2.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

**3.PROGRAM:**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
/* Defining equation to be solved.
   Change this equation to solve another problem. */
#define    f(x)    3*x - cos(x) -1
/* Defining derivative of g(x).
   As you change f(x), change this function also. */
#define   g(x)   3 + sin(x)

void main(){
        float x0, x1, f0, f1, g0, e;
        int step = 1, N;
        //clrscr();
    /* Inputs */
        printf("\nEnter initial guess:\n");
        scanf("%f", &x0);
        printf("Enter tolerable error:\n");
        scanf("%f", &e);
        printf("Enter maximum iteration:\n");
        scanf("%d", &N);
        /* Implementing Newton Raphson Method */
        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();
}
```

**4.OUTPUT:**

Enter initial guess:

0

Enter tolerable error:

4

Enter maximum iteration:

10

| Step | x0 | f(x0) | x1 | f(x1) |
|---|---|---|---|---|
| 1 | 0.000000 | -2.000000 | 0.666667 | 0.000000 |

Root is: 0.666667

# PROGRAM NO. 5

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech (CSE)

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

## 1.OBJECTIVE:

Write a program to find solution of any non polynomial equation using Iteration Method.

## 2.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

**3.PROGRAM:-**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
/* Define function f(x) which is to be solved */
#define   f(x)   cos(x)-3*x+1
/* Write f(x) as x = g(x) and define g(x) here */
#define   g(x) (1+cos(x))/3
int main(){
        int step=1, N;
        float x0, x1, e;
        //clrscr();
        /* Inputs */
       printf("Enter initial guess: ");
        scanf("%f", &x0);
        printf("Enter tolerable error: ");
        scanf("%f", &e);
        printf("Enter maximum iteration: ");
        scanf("%d", &N);
        /* Implementing Fixed Point Iteration */
        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);
}
```

**4. OUTPUT:**

Enter initial guess:

0

Enter tolerable error:

4

Enter maximum iteration:

10

| Step | x0 | f(x0) | x1 | f(x1) |
|---|---|---|---|---|
| 1 | 0.000000 | -2.000000 | 0.666667 | 0.000000 |

Root is: 0.666667

# PROGRAM NO. 6

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech (CSE)

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

**1.OBJECTIVE:**

Write a program to perform Gauss Elimination Method.

**2.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

**3.PROGRAM:**

```
#include<stdio.h>
#include<conio.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);}
```

**4.OUTPUT:**

Enter the order of matrix: 3

Enter the elements of augmented matrix row-wise:

A[1][1] : 1

A[1][2] : -2

A[1][3] : 3

A[1][4] : 0

A[2][1] : 3

A[2][2] : -2

A[2][3] : -5

A[2][4] : -4

A[3][1] :5

A[3][2] : -1

A[3][3] : -4

A[3][4] : 3

The solution is:

x1=1.840000

x2=2.360000

x3=0.960000

# PROGRAM NO. 7

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech (CSE)

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

**1.OBJECTIVE:**

Write a program to perform Gauss Jordon Elimination method .

**2.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

**3.PROGRAM:**

```
#include<stdio.h>
void solution( int a[][20], int var );
int main(){
    int a[ 20 ][ 20 ], var, i, j, k, l, n;
    printf( "\nEnter the number of variables:\n" );
    scanf( "%d", &var )
    for ( i = 0;i < var;i++ )
    {
        printf( "\nEnter the equation%d:\n", i + 1 );
        for ( j = 0;j < var;j++ )
        {
            printf( "Enter the coefficient of  x%d:\n", j + 1 );
            scanf( "%d", &a[ i ][ j ] );
        }
        printf( "\nEnter the constant:\n" );
        scanf( "%d", &a[ i ][ var] );
    }
     solution( a, var );
    return 0;
}
 void solution( int a[ 20 ][ 20 ], int var ){
    int k, i, l, j;
  for ( k = 0;k < var;k++ ){
        for ( i = 0;i <= var;i++ )
```

```
        {
            l = a[ i ][ k ];
            for ( j = 0;j <= var;j++ ) {
                if ( i != k )
                a[i][j] = (a[k][k]*a[i][j])-(l*a[k][j]);
            }
        }
    }

    printf( "\nSolutions:" );

    for ( i = 0;i < var;i++ )
    {
        printf( "\nTHE VALUE OF x%d IS %f\n", i + 1, ( float ) a[ i ][ var ] / ( float ) a[ i ][ i ] );
    }

}
```

**4. OUTPUT:**

Enter the number of variables: 3

Enter the equation 1:

Enter the coefficient of  x1: 1

Enter the coefficient of  x2: 0

Enter the coefficient of  x3: 0

Enter the constant: 2

Enter the equation 2:

Enter the coefficient of  x1: 0

Enter the coefficient of  x2: 1

Enter the coefficient of  x3: 0

Enter the constant: 0

Enter the equation 3:

Enter the coefficient of  x1: 0

Enter the coefficient of  x2: 0

Enter the coefficient of  x3: 1

Enter the constant: -1

Solutions:

THE VALUE OF x1 IS 2.000000

THE VALUE OF x2 IS 0.000000

THE VALUE OF x3 IS -1.000000

# PROGRAM NO. 8

**NAME :** Tejaswani Upadhyay

**COURSE /BRANCH :** BTech CSE

**SEM/SEC :** 5 / H

**ROLL NO :** 54

**DATE :**

**1.OBJECTIVE:**

Write a Program to perform Gauss Seidel Method.

**2.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

**3.PROGRAM:**

```
#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
 {
```

```
  /* Calculation */
  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);
  /* Error */
  e1 = fabs(x0-x1);
  e2 = fabs(y0-y1);
  e3 = fabs(z0-z1);
count++;
  /* Set value for next iteration */
  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;
}
```

**4. OUTPUT:**

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