LAB-1

OBJECTIVE 1:

Two types of measures are commonly used for error calculation.

- Absolute Error: Absolute error is the positive difference between the actual value and the approximate value of a variable. If x is the actual value and x_a is the approximation to x, then the absolute error is given by:

$$E_A=|x-x_a|$$

- Relative Error: Relative error is the ratio of the error to the actual value of a variable. If x is the actual value and x_a is the approximation to x, then the relative error is given by:

$$E_R=|(x-x_a)/x |$$

i. Write a program in "C" Language to deduce absolute and relative errors.

Program:

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
void main()
  {
   doubleabs_err, rel_err, p_rel_err, t_val, a_val;
   printf("\n INPUT TRUE VALUE:");
   scanf("%lf", &t_val);
   printf("\n INPUT APPROXIMATE VALUE:");
   scanf("%lf", &a_val);
   abs_err=fabs(t_val-a_val);
   rel_err=abs_err/t_val;
   p_rel_err=rel_err*100;
   printf("\nABSOLUTE ERROR= %lf", abs_err);
   printf("\nRELATIVE ERROR= %lf", rel_err);
   printf("\nPERCENTAGE RELATIVE ERROR= %lf", p_rel_err);
   getch();

   }
```

OBJECTIVE 2:

A polynomial equation is an equation that can be written in the form

$$ax^n+bx^{n-1}+….+rx+s=0,$$

where a,b,….r and s are constants. We call the largest exponent of x appearing in a non-zero term of a polynomial, the degree of that polynomial.

ii. Write a program in "C" Language to evaluate a given polynomial equation and then to deduce errors involved in it.

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        float a[100],sum=0,x;
        float approx;
        int n,i;
        clrscr();
        printf("Enter the degree of the polynomial");
        scanf("%d",&n);
        printf("Enter the coefficients into the array");
        for(i=n;i>=0;i--)
          {
             scanf("%f",&a[i]);
           }
        printf("Enter the value of x");
        scanf("%f",&x);
        for(i=n;i>0;i--)
         {
            sum=(sum+a[i])*x;
         }
        sum=sum+a[0];
        printf("\n value of the polynomial is=%f", sum);
        printf("Enter the approximate value: ");
        scanf("%f",&approx);
        printf("Absolute error=%f",(sum-approx));
        getch();
}
```

OBJECTIVE 3:

Truncation and Round-off: Truncation error occurs, when some digits from the number are discarded whereas rounding-off the number also causes truncation but with some adjustment to the last digit retained depending on the value of the truncated digits.

iii. Write a program in "C" Language to deduce errors upto 2 decimal digits by round-off and truncation.

LAB-2

OBJECTIVE 1:

Bisection method, is one of the simplest iterative methods. Two initial approximations, say x_0 and x_1 such that $f(x_0)*f(x_1)<0$ which ensures that the root lies between x_0 and x_1 are taken. The next x-value, say x_2, as the mid-point of the interval $[x_0, x_1]$ is computed.

i. Write a program in "C" Language that will find the root of the Algebraic equations using Bisection method.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define e 0.001
#define f(x) x*x*x-4*x-9
void main()
{
 float x0,x1,x2;
 float f0,f1,f2;
 int i=0;
 clrscr();
printf("Enter the values of x0 and x1");
scanf("%f %f", &x0, &x1);
do
{
            f0=f(x0);
            f1=f(x1);
            x2=(x0+x1)/2;
            f2=f(x2);
            if(f0*f2<0)
            {
                        x1=x2;
            }
          else
           {
                        x0=x2;
           }
          i++;
        printf("no of iterations %d", i);
        printf("root=%f", x2);
        printf("value of function=%f\n",f2);
      }while(fabs(f2)>e);
      getch();
     }
```

OBJECTIVE 2:

A non-polynomial equation is called transcendental equation. Some examples of transcendental equations are:

$2^x-x-3=0$ $e^x\cos x-3x=0$

ii. Write a program in "C" Language that will find the root of the Transcendental equations using Bisection method.

LAB-3

Though the bisection method guarantees that iterative process will converge, but its convergence is slow. The false position method, also known as regula-falsi or method of linear interpolation, is similar to the bisection method but faster than it. It also starts with two initial approximations to the root, say x_0 and x_1, for which *f(x)* has opposite signs, and then by linear interpolation the next approximation is determined.

OBJECTIVE 1:

i. Write a program in "C" Language that will find the root of the Algebraic equations using Regula-Falsi method.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define e 0.001;
#define f(x) x*x*x -2*x -5
void main()
{
        int i=0;
        float x0,x1,x2,f0,f1,f2;
        printf("enter the values of x0 and x1");
        scanf("%f%f",&x0,&x1);
        do
        {
            fo=f(x0);
            f1=f(x1);
           f2=f(x2);
           x2=((x0*f1)-(x1*f0))/(f1-f0);
         if( f0*f2<0)
        {
                x1=x2;
                f1=f2;
```

```
        }
        else
        {
                x0=x2;
                f0=f2;
        }
        i++;
        printf("No of iterations=%d\t",i);
        printf("root=%f\t",x2);
        printf("value of function=%f\n",f2);
}while(fabs(f2)>e);
getch();
}
```

OBJECTIVE 2:

ii. Write a program in "C" Language that will find the root of the Transcendental equations using Regula-Falsi method.