AIM: Write a program to find the root of an equation using **BISECTION METHOD.**

We use the following functions to perform bisection program:

- **initialize( ):** It stores the equation, whose roots are to be determined.
- **operation( ):** The operation function finds the interval in which the roots of the equation lie.
- **verify_roots( ):** It verifies the given interval(by the user) of the roots of the equation.
- **find_roots( ):** By using the interval of the roots(either entered by the user or determined by the program), the find_roots function finds the exact roots by taking '**allowed error and no. of iterations**' as argument from the user.
- **func( ):**This function finds the solution of the entered equation at a particular value.

BASIC FORMULA

$$x = \frac{(a+b)}{2}$$

PROGRAM

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


float func(float);
float find_root(float,float,float,int);
int verify_roots(float,float);
float operation(int);

/*=======MAIN Function=====*/
void main()
{
  int ch,k,iter,flag,check;
  float r1,r2,root,ae;
  clrscr();
  start:
  printf("\n======MENU====\n");
  printf("1) Enter the limits of the
roots of given polynomial\n");
  printf("2) Let the program calculate
the limits of the roots.\n");
  printf("Your choice: ");
  scanf("%d",&ch);
  switch(ch)
  {
    case 1:
      printf("\nEnter the Roots: ");
      scanf("%f %f", &r1, &r2);
      k=verify_roots(r1,r2);
      if(k==1)
      {
         printf("\nEnter the allowed
error and number of iterations: ");
         scanf("%f %d",&ae, &iter);
         root=find_root(r1,r2,ae,iter);
         printf(" the root is: %f",root);
      }
      else if (k==0)
```

```
      {
         goto start;
      }
      break;
   case 2:
       r1=operation(flag);
       printf("\nEnter the allowed
error and number of iterations: ");
       scanf("%f %d",&ae, &iter);
       root=find_root(r1,r1+0.5,ae,iter
);
       printf("the root is: %f",root);
       break;
   default:
       printf("Please Enter a valid
choice.\n");
       goto start;
  }
  getch();
}

/*=======find_root ( )=====*/
float find_root(float r1,float r2, float
ae, int n)
{
 int count=0;
 float k,avg_prev=0, avg=0, aer=0;
 if(n==0)
 {
   n=3;
   printf("\nBy default, 3 iterations
will be executed.");
   /*Three iterations from 0 to 2*/
 }
 if(ae==0)
 {
  ae=0.01;
  printf("\nBy default, 0.01 is set as
allowed error.");
 }
  printf("\n| #No\t| r1 \t| r2 \t| x
\t|f(x)\t\t\t| aer \t| ae \t|\n");

printf("|=======|=======|======
=|=======|=================
=====|=======|=======|\n");
 do
 {
  printf("|%3d\t|", ++count);
  printf("%0.5f|%0.5f|", r1,r2);
  avg_prev=avg;
  avg=(r1+r2)/2;
  printf("%0.5f|",avg);
  k=func(avg);
  printf(" %0.6f", k);
  if(k>0)
   {
     printf("\t(+ive)  |");
     r2=avg;
   }
   else if(k<0)
   {
    printf("\t(-ive)  |");
    r1=avg;
   }
   --n;
   if(n==-1)
   {
    n=0;
   }
  aer=fabs(avg-avg_prev);
```

```
/*
1) abs( ) is used to find the absolute
value(i.e only positive) of an
integer.
2) fabs( ) finds the absolute
value(only positive) of floating
numbers.
*/
```

```
   printf("%0.5f|%0.5f|\n", aer,ae);
 }
while(n!=0 || aer>ae);
```

/*This while statement will keep iterating unless any of one condition, i.e no. of iterations or allowed error both are satisfied.*/

```
   printf("\nAfter completing %d
iterations, ", count);
   return avg;
}
/*=======verify_roots ( )=====*/
int verify_roots(float r1, float r2)
{
 float k,l;
 k=func(r1);
 printf("|\tf(%f)=%f\t|\n ",r1, k);
 l=func(r2);
 printf("|\tf(%f)=%f\t|\n ",r2, l);
 if((k*l)>=0)
 {
   printf("\nThe actual root of the
polynomial do not lie between (%f,
%f).", r1,r2);
   return 0;
 }
 else if((k*l)<0)
 {
   printf("\n The entered values have
been tested. \nThe actual root lie
between (%f, %f)",r1,r2);
   return 1;
 }
 return 0;
}
/*=========func( )=========*/
float func(float x)
{
 float value=0;
 value=2*x*x*x-10*x*x-1;
 //value=x*log10(x)-1.2;
 //value=sin(x)-(1/x);
 return value;
}
/*=======operation ( )=====*/
float operation(int flag)
{
 float k,l,i;
 if(flag==0)
 {
   i=-5;
 }
 else
   i=0;
 k=func(i);
 printf("|f(%f)=%f ",i, k);
 if(k>0 || k==0)
   printf(" (+ive) \t|\n");
 else if(k<0)
   printf(" (-ive) \t|\n");
 l=func(i+=0.5);
 printf("|f(%f)=%f ",i, l);
 if(l>0 || l==0)
   printf(" (+ive) \t|\n");
 else if(l<0)
   printf(" (-ive) \t|\n");
 while((k*l)>=0)
  {
   i=i+0.5;
   k=l;
   l=func(i);
   printf("|f(%f)=%f ",i, l);
   if(l>0 || l==0)
     printf(" (+ive) \t|\n");
   else if(l<0)
     printf(" (-ive) \t|\n");
  }
  printf("\nThe roots lie between ( %f
, %f )", i-0.5,i);
 return (i-0.5);}
```

AIM: Write a program to find the root of an equation using **REGULA-FALSI METHOD.**

We use the following functions to perform bisection program:

- **initialize( ):** It stores the equation, whose roots are to be determined.
- **operation( ):** The operation function finds the interval in which the roots of the equation lie.
- **verify_roots( ):** It verifies the given interval(by the user) of the roots of the equation.
- **find_roots( ):** By using the interval of the roots(either entered by the user or determined by the program), the find_roots function finds the exact roots by taking '**allowed error and no. of iterations**' as argument from the user.
- **func( ):**This function finds the solution of the entered equation at a particular value.

BASIC FORMULA

$$x_2 = x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_0)$$

PROGRAM

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

float func(float);
float find_root(float,float,float,int);
int verify_roots(float,float);
float operation(int);

void main()
{
  int ch,k,iter,flag,check;
  float r1,r2,root,ae;
  clrscr();
  start:
  printf("\n======MENU====\n");
  printf("1) Enter the limits of the
roots of given polynomial\n");
  printf("2) Let the program calculate
the limits of the roots.\n");
  printf("Your choice: ");
  scanf("%d",&ch);
  switch(ch)
  {
    case 1:
       printf("\nEnter the Roots: ");
       scanf("%f %f", &r1, &r2);
       k=verify_roots(r1,r2);
       if(k==1)
       {
          printf("\nEnter the allowed
error and number of iterations: ");
          scanf("%f %d",&ae, &iter);
          root=find_root(r1,r2,ae,iter);
          printf("\n the root is:
%f",root);
       }
       else if (k==0)
       {
          goto start;
```

```
      }
      break;
   case 2:
        r1=operation(flag);
        printf("\nEnter the allowed
error and number of iterations: ");
        scanf("%f %d",&ae, &iter);
        root=find_root(r1,r1+0.5,ae,iter
);
        printf(" the root is: %f",root);
        break;
   default:
        printf("Please Enter a valid
choice.\n");
        goto start;
  }
  getch();
}
```

float find_root(float r1,float r2, float ae, int n)

```
{
 int count=0;
 float k,avg_prev=0, avg=0,
aer=0,term1=0,term2=0,term3=0,ter
m4=0,term5=0;
 if(n==0)
 {
   n=3;
   printf("\nBy default, 3 iterations
will be executed.");
   /*Three iterations from 0 to 2*/
 }
 if(ae==0)
 {
  ae=0.01;
  printf("\nBy default, 0.01 is set as
allowed error.");
 }
 printf("\n| #No\t| r1 \t| r2 \t| x
\t|f(x)\t\t\t| aer \t| ae \t|\n");
 printf("|=======|=======|======
=|=======|================
=====|=======|=======|\n");
 do
 {
  printf("|%3d\t|", ++count);
  printf("%0.5f|%0.5f|", r1,r2);
  avg_prev=avg;
  term1=r1*func(r2);
  term2=r2*func(r1);
  term3=term1-term2;
  term4=func(r2)-func(r1);
  avg= term3/term4;
  printf("%0.5f|",avg);
  k=func(avg);
  printf(" %0.6f", k);
  if(k>0)
   {
     printf("\t(+ive)  |");
     r2=avg;
   }
   else if(k<0)
   {
    printf("\t(-ive)  |");
    r1=avg;
   }
   --n;
   if(n==-1)
   {
    n=0;
   }
   aer=fabs(avg-avg_prev);
   printf("%0.5f|%0.5f|\n", aer,ae);
 }
 while(n!=0 || aer>ae);
 /*This while statement will keep
   iterating unless no. of iterations
   and allowed error both are
satisfied*/
```

```
  printf("\nAfter completing %d iterations, ", count);
  return avg;
}

int verify_roots(float r1, float r2)
{
 float k,l;
 k=func(r1);
 printf("|\tf(%f)=%f\t|\n ",r1, k);
 l=func(r2);
 printf("|\tf(%f)=%f\t|\n ",r2, l);
 if((k*l)>=0)
 {
  printf("\nThe actual root of the polynomial do not lie between (%f, %f).", r1,r2);
  return 0;
 }
 else if((k*l)<0)
 {
  printf("\n The entered values have been tested. \nThe actual root lie between (%f, %f)",r1,r2);
  return 1;
 }
 return 0;
}

float func(float x)
{
 float value=0;
 value=x*log10(x)-1.2;
 //value=x*log10(x)-1.2;
 //value=sin(x)-(1/x);
 //value=cos(x)-x*exp(x);
 return value;
}

float operation(int flag)
{
 float k,l,i;
 i=0;
 k=func(i);
 printf("|f(%f)=%f ",i, k);
 if(k>0 || k==0)
  printf(" (+ive) \t|\n");
 else if(k<0)
  printf(" (-ive) \t|\n");
 l=func(i+=0.5);
 printf("|f(%f)=%f ",i, l);
 if(l>0 || l==0)
  printf(" (+ive) \t|\n");
 else if(l<0)
  printf(" (-ive) \t|\n");
 while((k*l)>=0)
  {
  i=i+0.5;
  k=l;
  l=func(i);
  printf("|f(%f)=%f ",i, l);
  if(l>0 || l==0)
    printf(" (+ive) \t|\n");
  else if(l<0)
    printf(" (-ive) \t|\n");
  }
  printf("\nThe roots lie between ( %f , %f )", i-0.5,i);
 return (i-0.5);
}
```

AIM: Write a program to find the root of an equation using **NEWTON-RAPSON METHOD.**

We use the following functions to perform bisection program:

- **initialize( ):** It stores the equation, whose roots are to be determined.
- **operation( ):** The operation function finds the interval in which the roots of the equation lie.
- **verify_roots( ):** It verifies the given interval(by the user) of the roots of the equation.
- **find_roots( ):** By using the interval of the roots(either entered by the user or determined by the program), the find_roots function finds the exact roots by taking '**allowed error and no. of iterations**' as argument from the user.
- **func( ):**This function finds the solution of the entered equation at a particular value.
- **derivative( ):** This function finds the solution of derivative of the equation at any particular point.

BASIC FORMULA

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

PROGRAM

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

float derivative(float);
float func(float);
float find_root(float,float,float,int);
int verify_roots(float,float);
float operation();

void main()
{
  int ch,k,iter,flag,check;
  float r1,r2,root,ae;
  clrscr();
  start:
  printf("\n======MENU====\n");
  printf("1) Enter the limits of the
roots of given polynomial\n");
  printf("2) Let the program calculate
the limits of the roots.\n");
  printf("Your choice: ");
  scanf("%d",&ch);
  switch(ch)
  {
    case 1:
      printf("\nEnter the Roots: ");
      scanf("%f %f", &r1, &r2);
      k=verify_roots(r1,r2);
      if(k==1)
      {
         printf("\nEnter the allowed
error and number of iterations: ");
         scanf("%f %d",&ae, &iter);
         root=find_root(r1,r2,ae,iter);
         printf("\n the root is:
%f",root);
      }
      else if (k==0)
      {
```

```
            goto start;
        }
        break;
    case 2:
        r1=operation();
        printf("\nEnter the allowed
error and number of iterations: ");
        scanf("%f %d",&ae, &iter);
        root=find_root(r1,r1+0.5,ae,iter
);
        printf("\n the root is: %f",root);
        break;
    default:
        printf("Please Enter a valid
choice.\n");
        goto start;
  }
  getch();
}
float find_root(float r1,float r2,
float ae, int n)
{
 int count=0,count1=0;
 float x0,avg_prev=0,
prev_aer=0,avg=0,
aer=0,term1=0,term2=0,term3=0,ter
m4=0,term5=0;
 if(n==0)
 {
   n=3;
   printf("\nBy default, 3 iterations
will be executed.");
   /*Three iterations from 0 to 2*/
 }
 if(ae==0)
 {
  ae=0.01;
  printf("\nBy default, 0.01 is set as
allowed error.");
 }
 if(fabs(func(r1)-0)<fabs(func(r2)-0))
  {
    avg=r1;
    printf("\nLet us take x0 = %f\n",
r1);
  }
  else
  {
    avg=r2;
    printf("\nLet us take x0 = %f\n",
r2);
  }
 printf("\n|  #No\t|\tx\t|  aer \t|   ae
\t|\n");
printf("|=======|============
=|=======|=======|\n");
 do
 {
  printf("|%3d\t|", ++count);
  avg_prev=avg;
  avg=avg_prev-
(func(avg_prev)/derivative(avg_prev)
);
  printf("x%d = %0.5f\t|",count,avg);
   --n;
   if(n==-1)
   {
    n=0;
   }
   prev_aer=aer;
   aer=fabs(avg-avg_prev);
   printf("%0.5f|%0.5f|\n", aer,ae);
   if(aer==prev_aer && count1==3)
   {
     printf("Since we get the same
values of x, i.e %f, hence we stop
here, and donot go further.",avg);
     break;
   }
   if(count1==3)
   {
```

```
    count1=1;
   }
   count1++;
 }
 while(n!=0 || aer>ae);
   printf("\nAfter completing %d
iterations, ", count);
   return avg;
}

int verify_roots(float r1, float r2)
{
 float k,l;
 k=func(r1);
 printf("|\tf(%f)=%f\t|\n ",r1, k);
 l=func(r2);
 printf("|\tf(%f)=%f\t|\n ",r2, l);;
 if((k*l)>=0)
 {
   printf("\nThe actual root of the
polynomial do not lie between (%f,
%f).", r1,r2);
   return 0;
 }
 else if((k*l)<0)
 {
   printf("\n The entered values have
been tested. \nThe actual root lie
between (%f, %f)",r1,r2);
   return 1;
 }
 return 0;
}

float func(float x)
{
 float value=0;
 value=x*x*x*x-x-10;
 //value=x*log10(x)-1.2;
 //value=sin(x)-(1/x);
 //value=cos(x)-x*exp(x);
 return value;
}

float derivative(float x)
{
 float value=0;
 value=4*x*x*x-1;
 return value;
}

float operation()
{
 float k,l,i;
 i=0;
 k=func(i);
 printf("|f(%f)=%f ",i, k);
 if(k>0 || k==0)
   printf(" (+ive) \t|\n");
 else if(k<0)
   printf(" (-ive) \t|\n");
 l=func(i+=0.5);
 printf("|f(%f)=%f ",i, l);
 if(l>0 || l==0)
   printf(" (+ive) \t|\n");
 else if(l<0)
   printf(" (-ive) \t|\n");
 while((k*l)>=0)
  {
   i=i+0.5;
   k=l;
   l=func(i);
   printf("|f(%f)=%f ",i, l);
   if(l>0 || l==0)
     printf(" (+ive) \t|\n");
   else if(l<0)
     printf(" (-ive) \t|\n");
  }
  printf("\nThe roots lie between ( %f
, %f )", i-0.5,i);
 return (i-0.5);
}
```