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 takes the polynomial from the user and stores them in a linked list, pointed by a variable.
- **operation( ):** The operation function finds the interval in which the roots of the polynomial lie.
- **verify_roots( ):** It verifies the given interval(by the user) of the roots of the polynomial.
- **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 polynomial at a particular value.

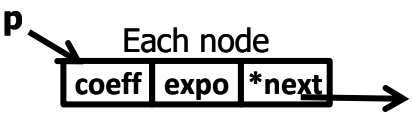
PROGRAM

```
#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);
int initialize();

struct node
{
 int coeff;
 int expo;
 struct node *next;
 };
struct node *p=NULL;

/*----------Main function----------*/
void main()
{
  int ch,k,iter,flag;
  float r1,r2,root,ae;
  clrscr();
  flag=initialize();
  printf("Initializated the polynomial
successfully!");
  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();
}
```

```
/*----------find_root function------*/
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.");
 }
 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)
   {
/*In case of recursion:
     find_root(avg,r2,ae,n-1);*/
      printf("\t(+ive)  |");
      r2=avg;
   }
   else if(k<0)
```

```
  {
/*In case of recursion:
    find_root(r1,avg,ae,n-1);*/
    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 function------*/
int verify_roots(float r1, float r2)
{
 float k,l;
 k=func(r1);
 printf("|f(%f)=%f \t|\n",r1, k);
 l=func(r2);
 printf("|f(%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 function------------*/
float func(float i)
{
 struct node *temp2=p;
 float value=0,value1=0;
 while(temp2!=NULL)
 {
  value1=pow(i,temp2->expo);
  value=value+(temp2->coeff)*value1;
  temp2=temp2->next;
 }
 return value;
}
```

```
/*--------operation function---------*/
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);
}
```

If the polynomial has no –ive term: (Ex: $5x^2+4x+10$) then the flag remains 0. For such equations we consider –ive roots, i.e f(-5), f(-4)…

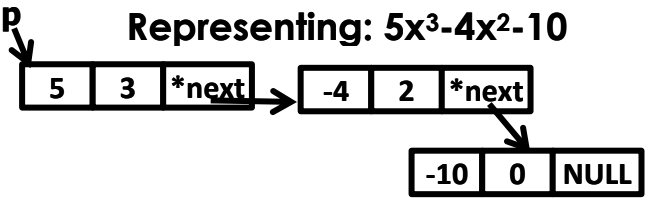
However, this is not applicable if the equation has large values for square indexes (Ex: $9x^2+x+10$; $7x^4+x^3+1$)

```
/*--------initialize function----------*/
int initialize()
{
 int c,e,flag=0;
 struct node *temp2;
 printf("Enter the expression: ");
 scanf("%dx^%d",&c,&e);
 while(c!=0 || e!=0)
 {
   struct node* temp=(struct
node*)malloc(sizeof(struct node));
   temp->coeff=c;
   if(c<0)
     flag=1;
   temp->expo=e;
   temp->next=NULL;

   if(p==NULL)
   {
     p=temp;
   }
   else
   {
     struct node *temp1=p;
     while(temp1->next!=NULL)
     {
      temp1=temp1->next;
     }
     temp1->next=temp;
   }
   scanf("%dx^%d",&c,&e);
 }
 return flag;
}
```

If the polynomial has even a single –ve term, then the flag turns 1 else if the polynomial has no –ive term then the flag remains 0.

Example:



OUTPUT-1

Case 1: When a positive root exists:

```
Enter the expression: 2x^3-10x^2-1x^0+0x^0
Initialized the polynomial successfully!
======MENU====
1) Enter the limits of the roots of given polynomial
2) Let the program calculate the limits of the roots.
Your choice: 2

|f(0.000000)=-1.000000  (-ive)          |
|f(0.500000)=-3.250000  (-ive)          |
|f(1.000000)=-9.000000  (-ive)          |
|f(1.500000)=-16.750000  (-ive)         |
|f(2.000000)=-25.000000  (-ive)         |
|f(2.500000)=-32.250000  (-ive)         |
|f(3.000000)=-37.000000  (-ive)         |
|f(3.500000)=-37.750000  (-ive)         |
|f(4.000000)=-33.000000  (-ive)         |
|f(4.500000)=-21.250000  (-ive)         |
|f(5.000000)=-1.000000  (-ive)          |
|f(5.500000)=29.250000  (+ive)          |

The roots lie between ( 5.000000 , 5.500000 )
Enter the allowed error and number of iterations: 0.00001 0

By default, 3 iterations will be executed.
|  #No  |   r1  |   r2  |   x   |f(x)                    |  aer  |   ae  |
|=======|=======|=======|=======|========================|=======|=======|
|  1    |5.00000|5.50000|5.25000| 12.781250     (+ive)  |5.25000|0.00001|
|  2    |5.00000|5.25000|5.12500| 5.566406     (+ive)  |0.12500|0.00001|
|  3    |5.00000|5.12500|5.06250| 2.203613     (+ive)  |0.06250|0.00001|
|  4    |5.00000|5.06250|5.03125| 0.582092     (+ive)  |0.03125|0.00001|
|  5    |5.00000|5.03125|5.01562| -0.213867    (-ive)  |0.01562|0.00001|
|  6    |5.01562|5.03125|5.02344| 0.182892     (+ive)  |0.00781|0.00001|
|  7    |5.01562|5.02344|5.01953| -0.015793    (-ive)  |0.00391|0.00001|
|  8    |5.01953|5.02344|5.02148| 0.083473     (+ive)  |0.00195|0.00001|
|  9    |5.01953|5.02148|5.02051| 0.033829     (+ive)  |0.00098|0.00001|
| 10    |5.01953|5.02051|5.02002| 0.009010     (+ive)  |0.00049|0.00001|
| 11    |5.01953|5.02002|5.01978| -0.003391    (-ive)  |0.00024|0.00001|
| 12    |5.01978|5.02002|5.01990| 0.002792     (+ive)  |0.00012|0.00001|
| 13    |5.01978|5.01990|5.01984| -0.000282    (-ive)  |0.00006|0.00001|
| 14    |5.01984|5.01990|5.01987| 0.001255     (+ive)  |0.00003|0.00001|
| 15    |5.01984|5.01987|5.01985| 0.000477     (+ive)  |0.00002|0.00001|
| 16    |5.01984|5.01985|5.01984| 0.000099     (+ive)  |0.00001|0.00001|
After completing 16 iterations, the root is: 5.019844
```

- **#No: No. of iterations**
- **r1: Lower interval of root**
- **r2: Higher interval of root**
- **x: average value of r1 and r2**
- **aer: current error**
- **ae: accepted error by the user**

OUTPUT-2

Case 2: When a negative root exists:

```
Enter the expression: 5x^3+4x^2+3x^1+10x^0+0x^0
Initializated the polynomial successfully!
======MENU====
1) Enter the limits of the roots of given polynomial
2) Let the program calculate the limits of the roots.
Your choice: 2
|f(-5.000000)=-530.000000  (-ive)       |
|f(-4.500000)=-378.125000  (-ive)       |
|f(-4.000000)=-258.000000  (-ive)       |
|f(-3.500000)=-165.875000  (-ive)       |
|f(-3.000000)=-98.000000  (-ive)        |
|f(-2.500000)=-50.625000  (-ive)        |
|f(-2.000000)=-20.000000  (-ive)        |
|f(-1.500000)=-2.375000  (-ive)         |
|f(-1.000000)=6.000000  (+ive)          |

The roots lie between ( -1.500000 , -1.000000 )
Enter the allowed error and number of iterations: 0.00001 0

By default, 3 iterations will be executed.
|  #No  |   r1   |   r2   |   x    |f(x)                  |  aer  |   ae  |
|=======|========|========|========|======================|=======|=======|
|  1    |-1.50000|-1.00000|-1.25000| 2.734375   (+ive)   |1.25000|0.00001|
|  2    |-1.50000|-1.25000|-1.37500| 0.439453   (+ive)   |0.12500|0.00001|
|  3    |-1.50000|-1.37500|-1.43750| -0.899170  (-ive)   |0.06250|0.00001|
|  4    |-1.43750|-1.37500|-1.40625| -0.213165  (-ive)   |0.03125|0.00001|
|  5    |-1.40625|-1.37500|-1.39062| 0.117260   (+ive)   |0.01562|0.00001|
|  6    |-1.40625|-1.39062|-1.39844| -0.046917  (-ive)   |0.00781|0.00001|
|  7    |-1.39844|-1.39062|-1.39453| 0.035430   (+ive)   |0.00391|0.00001|
|  8    |-1.39844|-1.39453|-1.39648| -0.005679  (-ive)   |0.00195|0.00001|
|  9    |-1.39648|-1.39453|-1.39551| 0.014892   (+ive)   |0.00098|0.00001|
| 10    |-1.39648|-1.39551|-1.39600| 0.004611   (+ive)   |0.00049|0.00001|
| 11    |-1.39648|-1.39600|-1.39624| -0.000534  (-ive)   |0.00024|0.00001|
| 12    |-1.39624|-1.39600|-1.39612| 0.002039   (+ive)   |0.00012|0.00001|
| 13    |-1.39624|-1.39612|-1.39618| 0.000753   (+ive)   |0.00006|0.00001|
| 14    |-1.39624|-1.39618|-1.39621| 0.000111   (+ive)   |0.00003|0.00001|
| 15    |-1.39624|-1.39621|-1.39622| -0.000212  (-ive)   |0.00002|0.00001|
| 16    |-1.39622|-1.39621|-1.39622| -0.000051  (-ive)   |0.00001|0.00001|

After completing 16 iterations, the root is: -1.396217
```

OUTPUT-3

Case 3: When user enters the roots

```
Enter the expression: 5x^3+4x^2+3x^1+10x^0+0x^0
Initializated the polynomial successfully!
======MENU====
1) Enter the limits of the roots of given polynomial
2) Let the program calculate the limits of the roots.
Your choice: 1
Enter the Roots: -1 -0.5
|       f(-1.000000)=6.000000   |
|       f(-0.500000)=8.875000   |

The actual root of the polynomial do not lie between (-1.000000, -
0.500000).

======MENU====
1) Enter the limits of the roots of given polynomial
2) Let the program calculate the limits of the roots.
Your choice: 1
Enter the Roots: -1.5 -1
|       f(-1.500000)=-2.375000  |
|       f(-1.000000)=6.000000   |

The entered values have been tested.
The actual root lie between (-1.500000, -1.000000)
Enter the allowed error and number of iterations: 0 15

By default, 0.01 is set as allowed error.
|  #No  |    r1  |   r2   |   x    |f(x)                  |  aer  |   ae  |
|=======|========|========|========|======================|=======|=======|
|  1    |-1.50000|-1.00000|-1.25000| 2.734375   (+ive)   |1.25000|0.01000|
|  2    |-1.50000|-1.25000|-1.37500| 0.439453   (+ive)   |0.12500|0.01000|
|  3    |-1.50000|-1.37500|-1.43750| -0.899170  (-ive)   |0.06250|0.01000|
|  4    |-1.43750|-1.37500|-1.40625| -0.213165  (-ive)   |0.03125|0.01000|
|  5    |-1.40625|-1.37500|-1.39062| 0.117260   (+ive)   |0.01562|0.01000|
|  6    |-1.40625|-1.39062|-1.39844| -0.046917  (-ive)   |0.00781|0.01000|
|  7    |-1.39844|-1.39062|-1.39453| 0.035430   (+ive)   |0.00391|0.01000|
|  8    |-1.39844|-1.39453|-1.39648| -0.005679  (-ive)   |0.00195|0.01000|
|  9    |-1.39648|-1.39453|-1.39551| 0.014892   (+ive)   |0.00098|0.01000|
| 10    |-1.39648|-1.39551|-1.39600| 0.004611   (+ive)   |0.00049|0.01000|
| 11    |-1.39648|-1.39600|-1.39624| -0.000534  (-ive)   |0.00024|0.01000|
| 12    |-1.39624|-1.39600|-1.39612| 0.002039   (+ive)   |0.00012|0.01000|
| 13    |-1.39624|-1.39612|-1.39618| 0.000753   (+ive)   |0.00006|0.01000|
| 14    |-1.39624|-1.39618|-1.39621| 0.000111   (+ive)   |0.00003|0.01000|
| 15    |-1.39624|-1.39621|-1.39622| -0.000212  (-ive)   |0.00002|0.01000|

After completing 15 iterations, the root is: -1.396225
```