Problem-1:

Determination of a root correct to three decimal places for an equation using Bisection Method and False-Position Method.

Theory:

According to bisection method, if a function f(x) is continuous between a and b, and f(a) and f(b) are of opposite signs, then there exits at least one root between a and b. If f(a) is negative and f(b) is positive, then the root lies between a and b and let its approximate value be given by $x_0 = (a + b)/2$. If $f(x_0) = 0$, we can say x_0 is a root of the equation. Otherwise root is between x_0 and b, or x_0 and a depending on whether $f(x_0)$ positive or negative. False position method is almost same as bisection method. This method is also known as regula falsi method. The difference between them is in false position method x_0 is calculated using the formula below,

$$x_0 = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

Code:

```
#include <bits/stdc++.h>
#include <conio.h>
using namespace std;
double a, b, c, d;
double valueCheck(double x)
{
    float ans = a * pow(x, 3) + b * pow(x, 2) + c * x + d;
    return ans;
}
int main()
{
    cout << "Equation is: ax^3+bx^2+cx+d=0\n";
    cout << "Enter the value of a, b, c and d: ";
    cin >> a >> b >> c >> d;
    int choice = -1;
    while (choice != 3)
    {
        system("cls");
        double i = 0;
        double x = 0;
        double y = 0;
        double prev=0;
        double root;
        int p = 0;
        int q = 0;
        cout << "1.Bisection Method\n2.False Position Method\n3.Exit\n";
        cout << "Enter the method: ";
        cin >> choice;
        while (true)
        {
            if (valueCheck(i) < 0 && p == 0)
            {
                x = i;
                p = 1;
            }
            else if (valueCheck(i) > 0 && q == 0)
            {
```

```
            y = i;
            q = 1;
        }
        double j = 0 - i;
        if (valueCheck(j) < 0 && p == 0)
        {
            x = j;
            p = 1;
        }
        else if (valueCheck(j) > 0 && q == 0)
        {
            y = j;
            q = 1;
        }
        if (p != 0 && q != 0)
        {
            break;
        }
        i++;
    }
    double Xr;
    int k = 1;
    while (true)
    {
        if (choice == 1)
        {
            Xr = (x + y) / 2;
        }
        else
        {
            double  f = valueCheck(x);
            double  g = valueCheck(y);
            Xr = (x * g - y * f) / (g - f);
        }
        if (fabs(Xr-prev) <= 0.0001)
        {
            break;
        }
        if (valueCheck(Xr) < 0)
        {
            x = Xr;
        }
        else if (valueCheck(Xr) > 0)
        {
            y = Xr;
        }
        else
        {
            break;
        }
        prev=Xr;
        cout << k << " | " << x << " | " << y << " | " << Xr << " | " << valueCheck(Xr) << " | " << fabs(valueCheck(Xr)) <<
endl;
        cout << "________________________________________________________" << endl;
        k++;
    }
    cout << "So root is: " << Xr << endl;
    cout << endl;
```

```
        getch();
    }
    return 0;
}
```

Output:

```
Equation is: ax^3+bx^2+cx+d=0
Enter the value of a, b, c and d: 1 0 -2 -5
1.Bisection Method
2.False Position Method
3.Exit
Enter the method: 1
1 | 1.5 | 3 | 1.5 | -4.625  | 4.625
_______________________________________________________
2 | 1.5 | 2.25  | 2.25 | 1.89062  | 1.89062
_______________________________________________________
3 | 1.875  | 2.25  | 1.875  | -2.1582  | 2.1582
_______________________________________________________
4 | 2.0625  | 2.25  | 2.0625  | -0.351318  | 0.351318
_______________________________________________________
5 | 2.0625  | 2.15625  | 2.15625  | 0.712799  | 0.712799
_______________________________________________________
6 | 2.0625  | 2.10938  | 2.10938  | 0.166836  | 0.166836
_______________________________________________________
7 | 2.08594  | 2.10938  | 2.08594  | -0.0956788  | 0.0956788
_______________________________________________________
8 | 2.08594  | 2.09766  | 2.09766  | 0.0347143  | 0.0347143
_______________________________________________________
9 | 2.0918  | 2.09766  | 2.0918  | -0.0306977  | 0.0306977
_______________________________________________________
10 | 2.0918  | 2.09473  | 2.09473  | 0.00195435  | 0.00195435
_______________________________________________________
11 | 2.09326  | 2.09473  | 2.09326  | -0.0143852  | 0.0143852
_______________________________________________________
12 | 2.09399  | 2.09473  | 2.09399  | -0.00621877  | 0.00621877
_______________________________________________________
13 | 2.09436  | 2.09473  | 2.09436  | -0.00213306  | 0.00213306
_______________________________________________________
14 | 2.09454  | 2.09473  | 2.09454  | -8.95647e-05  | 8.95647e-05
_______________________________________________________
So root is: 2.09464

1.Bisection Method
2.False Position Method
3.Exit
Enter the method: 2
1 | 0.714286  | 3 | 0.714286  | -6.06414  | 6.06414
_______________________________________________________
2 | 1.34249  | 3 | 1.34249  | -5.26542  | 5.26542
_______________________________________________________
3 | 1.7529  | 3 | 1.7529  | -3.11973  | 3.11973
_______________________________________________________
4 | 1.95639  | 3 | 1.95639  | -1.42479  | 1.42479
_______________________________________________________
5 | 2.04172  | 3 | 2.04172  | -0.572263  | 0.572263
_______________________________________________________
```

6	2.07481	3	2.07481	-0.217873	0.217873
7	2.08724	3	2.08724	-0.0812517	0.0812517
8	2.09185	3	2.09185	-0.0300673	0.0300673
9	2.09356	3	2.09356	-0.0110945	0.0110945
10	2.09419	3	2.09419	-0.00408939	0.00408939
11	2.09442	3	2.09442	-0.00150675	0.00150675

So root is: 2.0945

Conclusion:

From the output section we can see that, in both method the root is almost same. In bisection method, the root is 2.09464 and in false position method the root is 2.0945. But the matter of observation in this problem is that, in bisection method we needed 14 steps to calculate the root, but in false position method the number of steps are 11. So, we can say false position method is more efficient.