```cpp
/*

1. Write a program to count number of significant digits in a given
number.
*/
#include <iostream>
#include <string>
using namespace std;

int countDigit(string number) {
    bool decimalFound = false;
    string integerPart, decimalPart;

    // Split the number into integer and decimal parts
    for (char c : number) {
        if (c == '.') {
            decimalFound = true;
            continue;
        }
        if (!decimalFound) integerPart += c;
        else decimalPart += c;
    }

    // Remove leading zeros from integer part
    int start = 0;
    while (start < integerPart.size() && integerPart[start] == '0') {
        start++;
    }
    integerPart = integerPart.substr(start);

    // Handle numbers without decimal points (remove trailing zeros)
    if (!decimalFound) {
        int end = integerPart.size() - 1;
        while (end >= 0 && integerPart[end] == '0') {
            end--;
        }
        integerPart = integerPart.substr(0, end + 1);
    }
```

```
    // Handle numbers starting with decimal point (remove leading zeros
from decimal)
    if (integerPart.empty()) {
        int startDecimal = 0;
        while (startDecimal < decimalPart.size() &&
decimalPart[startDecimal] == '0') {
            startDecimal++;
        }
        decimalPart = decimalPart.substr(startDecimal);
    }

    // Count all remaining digits
    return integerPart.size() + decimalPart.size();
}

int main() {
    string num;
    cout << "Enter a number: ";
    cin >> num;

    int result = countDigit(num);
    cout << "Total significant digits: " << result << endl;

    return 0;
}
```

```
/*
C221050
2. Write a program to round off a number with n digits after decimal point
using banker's rule.
*/

#include<bits/stdc++.h>
using namespace std;

string roundNum(string num, int d) {
    int p = num.find('.');
```

```
    if (p == -1 || p + d + 1 >= num.length()) return num;

    char next = num[p + d + 1];
    char last = num[p + d];

    if (next < '5' || (next == '5' && (last - '0') % 2 == 0)) {
        return num.substr(0, p + d + 1);
    }

    for (int i = p + d; i >= 0; i--) {
        if (num[i] == '.') continue;
        if (num[i] < '9') {
            num[i]++;
            return num.substr(0, p + d + 1);
        }
        num[i] = '0';
    }

    return "1" + num.substr(0, p + d + 1);// (9.999 → 10.00), add '1' at
the start
}

int main() {
    string num;
    int d;

    cout << "Enter number: ";
    cin >> num;
    cout << "Decimal places: ";
    cin >> d;
    cout << "Rounded: " << roundNum(num, d) << endl;

    return 0;
}
```

```
/*

3.Write a program to evaluate a polynomial f(x) = x3 - 2x2 + 5x + 10  by
using Horner's rule x = 5.
*/



#include<bits/stdc++.h>
using namespace std;

int main()
{
    cout << "Enter the value of n = ";
    int n,x;
    cin >> n;

    vector<int> a(n+5),p(n+5);

    for(int i = n; i >= 0; i--){
        cout << "a[" << i << "] = ";
        cin >> a[i];
    }

    cout << "Value of x = ";

    cin >> x;

    p[n+1] = 0;

    for(int i = n; i >= 0; i--){
        p[i] = (p[i+1] * x) + a[i];
    }

    cout << "The answer is = " << p[0] << endl;

    return 0;

}
```

```cpp
/*

4.Write a program to find the root of the equation x3 - 9x + 1 = 0,
correct to 3 decimal places, by using the bisection method.

*/
#include<bits/stdc++.h>
using namespace std;

double error = .0001;

bool calc(double x)
{
    double xx = ( (x * x * x) - (9.0 * x) + 1.0 );
    if(xx < 0.0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

int main()
{
    double right = 3.0, left = 1.0;

    while(fabs(right - left) >= error)
    {
        double mid = (right + left) / 2.0;
        if(calc(mid)){
            left = mid;
        }
        else right = mid;
    }

    cout << fixed << setprecision(3) << left << endl;

    return 0;
```

```
}
```

```
/*

 5.Write a program to find all the roots of the equation x3 - 6x + 4 = 0,
correct to 3 decimal places. [Use bisection method].
*/

#include <bits/stdc++.h>
using namespace std;
double error = 0.0001;

double fun(double x)
{
    return 1.00 * ( (x * x *x) - (6 * x) + 4);
}

double bisection_root(double x1, double x2)
{
    while(fabs(x1 - x2) > error)
    {
        double mid = ( x1 + x2) / 2.00;
        if(fun(mid) * fun(x1) < 0.0)
        {
            x2 = mid;
        }

        else
        {
            x1 = mid;
        }

    }
    return x2;
}
```

```
int main()
{
    double lower = -100, upper = 100, x = 1.0;

    double x2 = lower, x1 = lower;

    while(x2 < upper)
    {
        x1 = lower;
        x2 = lower + x;
        double f1 = fun(x1);
        double f2 = fun(x2);
        lower = x2 + 0.1;

        if((f1 * f2) > 0)
        {
            continue;
        }
        double ans=bisection_root(x1, x2);

        cout << fixed <<setprecision(3)<<ans<< endl;
    }
}
```

```
/*

6.Write a program to find the root of the equation x3 - 6x + 4 = 0,
correct to 3 decimal places, by using Newton-Raphson method.
*/

#include<bits/stdc++.h>
using namespace std;


double error = .005;
```

```
double f_x1(double x1) // Computing f(x1)
{
    return (x1 * x1 * x1) - (6 * x1) + 4;
}


double f_prime(double x1) // Computing f'(x1)
{
    return (3 * x1 * x1) - 6.0;
}
int main()
{
    double x1 = 0;
    double x2 = x1 - ( f_x1(x1) / f_prime(x1) );
    while( fabs(x2 - x1) > error )
    {
       x1 = x2;
       x2 = x1 - ( f_x1(x1) / f_prime(x1) );
    }

    cout << "The result is: " << fixed << setprecision(4) << x1 << endl;

    return 0;
}
```

```
/*

7.Write a program to find the root of the equation x3 - x + 2 = 0, correct
to 3 decimal places, by using false position method.
*/

#include<bits/stdc++.h>
using namespace std;


double error = .00005;
```

```
double f_x(double x)
{
    return ((x * x * x) - x + 2) * 1.00;
}

int main()
{
    double x1 = -2.0, x2 = 1.0;

    double x0 = x1 - ( ( f_x(x1) * (x2- x1) ) / ( f_x(x2) - f_x(x1) ) );

    if( f_x(x1) * f_x(x0) < 0.0 )
    {
        x2 = x0;
    }
    else
    {
        x1 = x0;
    }

    double x0_prev = x0;

    x0 =x1-((f_x(x1)*(x2- x1))/(f_x(x2)-f_x(x1)));

    while( abs(x0_prev - x0) > error )
    {
        if( f_x(x1) * f_x(x0) < 0.0 )
        {
            x2 = x0;
        }
        else
        {
            x1 = x0;
        }

        x0_prev = x0;
        x0 = x1 - ( ( f_x(x1) * (x2- x1) ) / ( f_x(x2) - f_x(x1) ) );
    }

    cout << fixed << setprecision(3) << x0 << endl;
```

```
    return 0;
}
```

```
/*

8.Write a program to find the root of the equation x3 - 5x2 - 29 = 0,
correct to 3 decimal places, by using secant method.
*/

#include<bits/stdc++.h>
using namespace std;

double error = 0.0001;

double fx(double x)
{
    return ( (x * x * x) - (5 * x * x) - 29 );
}

double calc(double x0, double x1)
{
    return x1 - ( ( (x1 - x0) / (fx(x1) - fx(x0)) ) * fx(x1) );
}


int main()
{
    double x0 = 1, x1 = 6;
    while(fabs(x0 - x1) > error)
    {
        double x2 = calc(x0,x1);
        x0 = x1;
        x1 = x2;
    }
```

```
    cout << fixed << setprecision(3) << x1 << endl;

    return 0;
}
```

```
/*

9.Write a program to find the quotient polynomial q(x) such that p(x) = (x
- 2) q(x) where the polynomial p(x) = x3 - 5x2 + 10x - 8 = 0 has a root at
x = 2.
*/

#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n, x;
    cout << "Enter the degree of polynomial (n): ";
    cin >> n;
    cout << "Enter the root value (x): ";
    cin >> x;
    vector<int>a(n+5),b(n+5);

    cout << "Enter the coefficients of the polynomial:\n";

    for(int i = n-1; i>=0; i--)
    {
        cout << "a[" << i << "] = ";
        cin >> a[i];
    }
    b[n-1] = 0;
    for(int i = n-2; i >= 0; i--){
        b[i] = a[i+1] + (x * b[i+1]);
    }

    cout << "\nThe Quotient Polynomial q(x) is, \n";
```

```
    for(int i = n-2; i >= 0; i--){
        if(b[i] == 0){
            continue;
        }
        if(i == n-2){
            cout << b[i] << "x^" << i << " ";
            continue;
        }
        if(i == 0){
            if(b[i] >= 0) cout << "+ " << b[i] << " ";
            else cout << "- " << abs(b[i]) << " ";
            continue;
        }
        else{
            if(b[i] >= 0 ) cout << "+ "  << b[i] << "x^" << i << " ";
            else cout << "- " << abs(b[i]) << "x^" << i << " ";
        }
    }
    cout << "= 0" << endl;

    return 0;
}
```