আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম
الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

Lab Assignment - 1

Course Title: Numerical Methods Lab
Course Code: CSE - 4746

Submitted to:	Submitted by:
Prof. Shamsul Alam **Professor,** **Dept. of CSE.**	**Arnab Barua** **ID:C201010** **7AM** **Dept. of CSE**

1. **Write a program to count the number of significant digits in a given number.**
2. **Write a program to round off a number with n significant figures using banker's rule.**

Code:

```
/*

 Author : Catalyst71
 < While there is a code, there is a bug > ¯\_(ツ)_/¯

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


bool checkOk(int significantNumber, int givenPoint) {
    return givenPoint < significantNumber;
}


int main()
{
    string number, finalNumber;
    int significant_count = 0, lastSigPos = -1, val, pos1, pos2;
    bool dPoint = true;

    cout << '\t' << "Bankers Rounding Rule\n\n";
    cout << "Enter Number : ";
    cin >> number;
    cout << "Enter upto how many significant digits you want to
round up: ";
    cin >> val;

    for (int i = 0; i < number.size(); i++) {
        if (number[i] == '.') dPoint = false;
    }
    if (dPoint) {
        int temp = 0 ;
        int j = number.size() - 1;
        while (number[j] == '0')temp++, j--;
        significant_count = number.size() - temp;
        if (checkOk(significant_count, val) == false) {
```

```
            while (1) {
                cout << "Enter a different value which is less
than " << significant_count << " : ";
                cin >> val;
                if (checkOk(significant_count, val))break;
            }
        }
        if (number[val] > '5') number[val - 1] += 1;
        else if (number[val] == '5') {
            int digit = number[val - 1] - '0';
            if (digit & 1)number[val - 1] += 1;
        }
        for (int i = 0; i < val; i++) {
            finalNumber += number[i];
        }
        cout << "Rounded Value : " << finalNumber << "\n";
    }
    else {
        for (int i = 0; i < number.size(); i++) {
            if (number[i] >= '1' && number[i] <= '9') {
                significant_count++;
                lastSigPos = i;
            } else if (number[i] == '0') {
                if (lastSigPos == -1)continue;
                else {
                    significant_count++;
                    lastSigPos = i;
                }
            }
        }
        cout << significant_count << "\n";
        if (checkOk(significant_count, val) == false) {
            while (1) {
                cout << "Enter a different value which is less
than " << significant_count << " : ";
                cin >> val;
                if (checkOk(significant_count, val))break;
            }
        }
        significant_count = 0;
        lastSigPos = -1;
        val++;
```

```
        for (int i = 0; i < number.size(); i++) {
            if (number[i] >= '1' && number[i] <= '9') {
                significant_count++;
                lastSigPos = i;
            } else if (number[i] == '0') {
                if (lastSigPos != -1) {
                    significant_count++;
                    lastSigPos = i;
                }
            }
            if (significant_count == val - 1) {
                pos1 = lastSigPos;
            } else if (significant_count == val) {
                pos2 = lastSigPos;
                break;
            }
        }

        if (number[pos2] > '5') number[pos1] += 1;
        else if (number[pos2] == '5') {
            int digit = number[pos1] - '0';
            if (digit & 1)number[pos1] += 1;
        }
        for (int i = 0; i < pos2; i++) {
            finalNumber += number[i];
        }

        cout << finalNumber << "\n";
    }


}
```

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()
{
    /// Given  f(x) = x3 - 2x2 + 5x + 10  by using Horner's rule x
= 5.
    int x = 5, n = 3;
    int a[4] = {10,5,-2,1};
    int p[4];
    p[3] = a[3];
    for(int i = 2; i >=0; i--){
        p[i] = p[i+1] * x + a[i];
    }
    cout << "Solution of the equation is : " << p[0] << "\n";
}
```

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

Algorithm:

Algorithm: Bisection method

1. Decide initial values for x_1 and x_2 and stopping criterion E.
2. Compute $f_1 = f(x_1)$ and $f2 = f(x_2)$.
3. If $f_1 * f_2 > 0$, x_1 and x_2 do not bracket any root and go to step 1.
4. Compute $x_0 = (x_1 + x_2) / 2$ and compute $f_0 = f(x_0)$.
5. If $f_1 * f_0 < 0$ then set $x_2 = x_0$ else set $x_1 = x_0$.
6. If absolute value of $(x_2 - x_1)$ is less then E, then root = $(x_1 + x_2) / 2$ and go to step 7
   Else go to step 4
7. Stop.

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define E 0.0005
double intial_guess(double a2,double a1, double a0)
{
    return sqrt(((a1/a0)*(a1/a0)) - (2 * (a2/a0)));
}
double eqn(double x)
{
    return (x*x*x - 9*x + 1);
}
int main()
{
    /// Given equation : x^3 +0*x^2 - 9*x + 1 = 0
    double x_max = intial_guess(1,0,-9);
    double x1 = x_max;
    double x2 = x_max * -1;
    double f1 = eqn(x1);
    double f2 = eqn(x2);
    while(abs(x2-x1) >= E)
    {
        double x = (x1 + x2)/2;
        f1 = eqn(x1);
        f2 = eqn(x2);
        if(f1*f2 <0)
            x2 = x;
        else
            x1 = x;
    }
    double root = (x1 + x2)/2;
    cout << "Root " << root << "\n";
}
```

5. **Write a program to find the root of the equation $x^5 + 3x^2 - 10 = 0$, correct to 3 decimal places, by using the fixed point method.**

Algorithm:

Fixed Point method

To find the root of the equation $f(x) = 0$, we rewrite this equation in this way

$$x = g(x)$$

Let x_0 be an approximate value of the desire root. Substituting it for x as the right side of the equation, we obtain the first approximation $x_1 = g(x_0)$. Further approximation is given by $x_2 = g(x_1)$. This iteration process can be expressed in general form as

$$x_{i+1} = g(x_i) \quad i = 0, 1, 2, \ldots.$$

which is called the *fixed point iteration formula*. The iteration process would be terminated when two successive approximations agree within some specified error.

- This method of solution is also known as the *method of successive approximations* or *method of direct substitution*.

Example: Locate the root of the equation $f(x) = x^3 + x^2 - 1 = 0$.
Solution: The given equation can be expressed as $x = 1 / \sqrt{(x+1)}$.
Let us start with an initial value of $x_0 = 1$.

$x_1 = 0.7071$ $x_2 = 0.7654$ $x_3 = 0.7526$ $x4 = 0.7554 \ldots$

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define E 0.0005
double eqn(double x)
{
    return sqrt(10.0 / (x*x*x + 3.0));
}
int main()
{
    /// f(x) = x^5 + 3*x^2 - 10 = 0
    /// g(x) = sqrt(10/x^3+3)
    /// The iteration process would be terminated when two
successive approximations agree within some specified error.
    double x = 1.0, root,  pre_diff = 0;
    while(1)
    {
        double x1 = x;
        double x2 = eqn(x);
        if(abs(x2 - x1)<=E)
        {
            root = x2;
            break;
        }
        x = x2;
```

```
    }
    cout << root << "\n";
}
```

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

Algorithm:

Algorithm: Newton-Raphson Method

1. Assign an initial value for x, say x_0 and stopping criterion E.
2. Compute $f(x_0)$ and $f'(x_0)$.
3. Find the improved estimate of x_0
   $x_1 = x_0 - f(x_0) / f'(x_0)$
4. Check for accuracy of the latest estimate.
   If $|x_1 - x_0| < E$ then stop; otherwise continue.
5. Replace x_0 by x_1 and repeat steps 3 and 4.

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define E 0.0005
double eqn(double x)
{
    return (pow(x, 3) - 6 * x + 4);
}
double derivative_eqn(double x)
{
    return (3 * pow(x,2) - 6);
}

int main()
{
    double x1, x2, x = 0;
    while(1)
    {
        x1 = x - eqn(x)/derivative_eqn(x);
        if(abs(x1 - x) <= E)
            break;
        x = x1;
    }
```

```
    cout << "Root : " << x << "\n";

}
```

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

Algorithm:

False Position Algorithm

1. Decide initial values for x_1 and x_2 and stopping criterion E.
2. Compute $x_0 = x_1 - (f(x_1)\,(x_2 - x_1)) / (f(x_2) - f(x_1))$
3. If $f(x_0) * f(x_1) < 0$ set $x_2 = x_0$ otherwise set $x_1 = x_0$
4. If the absolute difference of two successive x_0 is less then E, then root = x_0 and stop. Else go to step 2.

Code:

```
///False Position Method - C201010

#include<bits/stdc++.h>
using namespace std;
#define E 0.0005
double intial_guess(double a2,double a1, double a0)
{
    return sqrt(((a1/a0)*(a1/a0)) - (2 * (a2/a0)));
}
double eqn(double x)
{
    return (x*x*x - x + 2);
}
int main()
{
    /// Given equation : x^3 +0*x^2 - 1*x + 2 = 0
    double x_max = intial_guess(1,0,-1);
    double x1 = x_max;
    double x2 = x_max*-1;
    double f1,f2,x, x_prev=0;
    while (1)
    {

        f1 = eqn(x1);
        f2 = eqn(x2);
        x = x1 - (f1*(x2-x1))/(f2-f1);
```

```
///         cout << f1 <<" "<< f2 << " : " << eqn(x) << "\n";
        if(eqn(x1)*eqn(x) <0)
            x2 = x;
        else
            x1 = x;
        if(abs(x_prev-x) < E) break;
        x_prev = x;

    }
    cout << "Root : " << x << "\n";

}
```

8. Write a program to find the root of the equation $x^3 - 5x^2 - 29 = 0$, correct to 3 decimal places, by using the secant method.

ALgorithm:

Algorithm: Secant Method

1. Decide two initial points x_1 and x_2 and required accuracy level E.
2. Compute $f_1 = f(x_1)$ and $f_2 = f(x_2)$
3. Compute $x_3 = (f_2 x_1 - f_1 x_2) / (f_2 - f_1)$
4. If $|x_3 - x_2| > E$, then
   set $x_1 = x_2$ and $f_1 = f_2$
   set $x_2 = x_3$ and $f_2 = f(x_3)$
   go to step 3
   Else
   set root = x_3
   print results
5. Stop.

Code:

```
///Secant Method - C201010

#include<bits/stdc++.h>
using namespace std;
#define E 0.0005
double intial_guess(double a2,double a1, double a0)
{
    return sqrt(((a1/a0)*(a1/a0)) - (2 * (a2/a0)));
}
double eqn(double x)
```

```
{
    return (x*x*x - 5*x*x - 29);
}
int main()
{
    /// Given equation : x^3 - 5*x^2 - 29 = 0
    double x_max = intial_guess(1,-5,0);
    double x1 = 2.0; //x_max;
    double x2 = 3.0; //x_max*-1;
    double f1,f2,x3, root;
    f1 = eqn(x1);
    f2 = eqn(x2);
    while (1)
    {
        x3 = (f2*x1 - f1*x2) / (f2-f1);
        if(abs(x3-x2) > E)
        {
            x1 = x2;
            f1 = f2;
            x2 = x3;
            f2 = eqn(x3);
        }
        else
        {
            root = x3;
            break;
        }
    }
    cout << "Root : " << root << "\n";

}
```

9. Write a program to find the *quotient polynomial* q(x) such that p(x) = (x - 2) q(x) where the polynomial $p(x) = x^3 - 5x^2 + 10x - 8 = 0$ has a root at x = 2.
Algorithm:

Example: The polynomial $p(x) = x^3 - 7x^2 + 15x - 9 = 0$ has a root at $x = 3$. Find the quotient polynomial $q(x)$ such that $p(x) = (x - 3)\, q(x)$.

Solution: Here, $a_3 = 1$, $a_2 = -7$, $a_1 = 15$, $a_0 = -9$

$b_3 = 0$

$b_2 = a_3 + b_3 * 3 = 1 + 0 = 1$

$b_1 = a_2 + b_2 * 3 = -7 + 3 = -4$

$b_0 = a_1 + b_1 * 3 = 15 + (-12) = 3$

Thus the polynomial $q(x)$ is

$x^2 - 4x + 3 = 0$

Code:

```
///Quotient Polynomial - C201010

#include<bits/stdc++.h>
using namespace std;
#define E 0.0005
int main()
{
    /// Given equation : x^3 - 5*x^2 + 10*x - 8 = 0
    int n = 4, root = 2;
    int a[n]= {-8,10,-5,1};
    int b[n];
    b[n-1] = 0;
    for(int i = n-2 ; i >=0 ; i--)
    {
        b[i] = a[i+1] + b[i+1]*root;
    }
    cout << b[n-2] << "x^2" << (b[n-3]>0?"+":" ") <<  b[n-3] <<
"x" << (b[n-4]>0?"+ ":" ") <<  b[n-4] << "\n";
}
```

10. Write a program to find all the roots of the equation $x^3 - 6x + 4 = 0$, correct to 3 decimal places.

Algorithm:

Algorithm

1. Choose lower limit **a** and upper limit **b** of the interval covering all the roots.
2. Decide the size of the increment interval Δx
3. set $x_1 = a$ and $x_2 = x_1 + \Delta x$
4. Compute $f_1 = f(x_1)$ and $f_2 = f(x_2)$
5. If $(f_1 * f_2) > 0$, then the interval does not bracket any root and go to step 9
6. Compute $x_0 = (x_1 + x_2)/2$ and $f_0 = f(x_0)$
7. If $(f_1 * f_2) < 0$, then set $x_2 = x_0$
   Else set $x_1 = x_0$ and $f_1 = f_0$
8. If $|x_2 - x_1| < E$, then
   root = $(x_1 - x_2) / 2$
   write the value of root
   go to step 9
   Else
   go to step 6
9. If $x_2 < b$, then set $a = x_2$ and go to step 3
10. Stop.

Code:

```
///All Possible Roots - C201010

#include<bits/stdc++.h>
using namespace std;
#define E 0.0005
double eqn(double x)
{
    return (x*x*x - 6*x + 4);
}
int main()
{
    /// Given equation : x^3 - 6x + 4 = 0
    double a = -100.0,b = 100.0;
    double delX = 0.00001;
    double x1,x2,f1,f2,x0,f0,root;

    while(1)
    {
        x1 = a;
        x2 = x1 + delX;
        f1 = eqn(x1);
        f2 = eqn(x2);
        if(f1 * f2 > 0)
        {
```

```
            if(x2 < b)
            {
                a = x2;
            }
            else break;
        }
        else
        {
            while(1)
            {
                x0 = (x1+x2)/2;
                f0 = eqn(x0);
                if(f1 * f0 < 0) x2 = x0;
                else
                {
                    x1 = x0;
                    f1 = f0;
                }
                if(abs(x2-x1) < E)
                {
                    root = (x1+x2)/2;
                    cout << "Root : " <<root << "\n";
                    if(x2 < b)
                    {
                        a = x2;
                    }
                    break;
                }
            }
        }
    }
    return 0;
}
```