UCCD1004 PROGRAMMING CONCEPTS AND PRACTICES

ASSIGNMENT 1

DUE DATE FOR ASSIGNMENT 1: WEEK 6, TUESDAY BEFORE 11:55PM

GENERAL INSTRUCTIONS

READ this assignment several times before starting, and FOLLOW all the instructions carefully.

1. EQUIPMENT

Hardware: Any suitable machines for completing the assignment.
Software: The Microsoft Visual Studio 2022. Your programming will be tested by the markers using Visual Studio 2022 or more updated versions.

2. PROJECT:

This is a **GROUP** assignment. Each group will have two students. NO help should be obtained from any persons outside the group. Do not seek help from the Internet. Do not post the assignment onto any social media.

3. DESCRIPTION OF PROJECT:

The assignment focuses on some important aspects of structured programming practices. The aim of this assignment is to provide an opportunity to demonstrate that you have acquired the skills and ability to be proficient in the area of developing C++ application program using modular approach.

4. ACADEMIC HONESTY AND COLLABORATION

Cooperation is recommended ONLY in understanding various concepts and system features. However, the actual solution of the assignment, the programming and debugging must be your group work, except for what you specifically credit to other sources. Your grade will be based on your own contribution. For example, copying without attribution any part of someone's program is plagiarism, even if you modify it and even if the source is a textbook. You can document the credit to other sources at the start of your program code listing. Offenders will be awarded with zero mark for this assignment.

5. PENALTY

Penalties such as downgrade or reduction of marks will be given to empty or late submissions unless legitimate reasons are provided.

DETAILS

A. Objective

Write a C++ programming for accepting a cubic equation as input, outputting the number of real solutions, and outputting the solution when the number of real solutions is one.

B. Terminologies

Complex number is not within our scope. For simplicity, the word *solution*, as well as all numbers, are referred to *real* values in this assignment.

Two equal solutions are considered as a single one; and by two (resp. three) solutions, we mean two (resp. three) distinct solutions.

A cubic equation has the form $a_3x^3 + a_2x^2 + a_1x + a_0 = 0$, where $a_3, a_2, a_1,$ and a_0, are all coefficients, where $a_3 \neq 0$. We assume $a_3 > 0$ without loss of generality. (Why?)

For convenience, let $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Now, $y = f(x)$ is a *curve* in the x-y plane. For avoiding ambiguity, curve is referred to $y = f(x)$, and the term *equation* is only used for $f(x) = 0$, in this assignment.

$f'(x)$ is the first derivative of $f(x)$.

C. Background Knowledge

Solving $f(x) = 0$ for x is equivalent to finding the points where the curve $y = f(x)$ cuts or touches the x-axis. The number of such points varies from one to three, depending on the actual values of $a_3, a_2, a_1,$ and a_0. We classify all cases into five categories as follows:

Category 1
The curve has two stationary points and one inflection point, and the stationary points are both positive or both negative. Since there is one cutting point in the x-axis, there is only one solution for $f(x) = 0$. Figure 1 is an example.

Category 2
The curve has two stationary points and one inflection point, and one of the stationary points is zero. Then, the curve cuts the x-axis once, and touches it once. So, there are two solutions for $f(x) = 0$. Figure 2 shows an example.

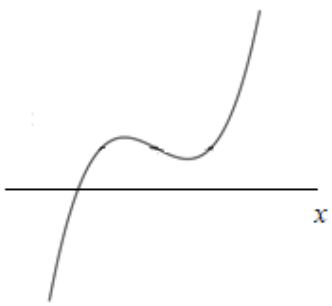
Category 3
The curve has two stationary points and one inflection point, and the stationary points are on opposite sides about the x-axis. That is, one has positive y-value, and the other one's y-value is negative. The curve cuts the x-axis three times. Therefore, there are three solutions for $f(x) = 0$. Figure 3 is an example.

Category 4

The stationary points and the infection point in the curve $y = f(x)$ will be located at the same point. Since there is either one cutting point or one touching point in the x-axis, there is only one solution for $f(x) = 0$. Figure 4 is an example.

Category 5

There is one infection point, but no stationary points, in the curve $y = f(x)$. The curve cuts the x-axis once, and therefore, there is only one solution for $f(x) = 0$. Figure 5 is an example.



Figures 1

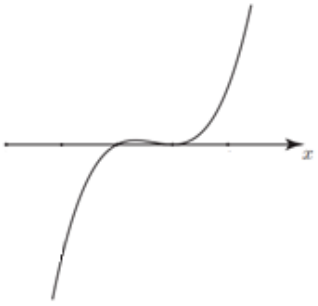


Figure 2

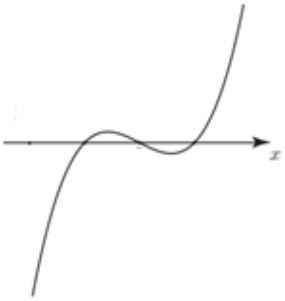


Figure 3

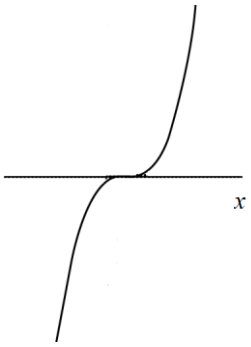


Figure 4

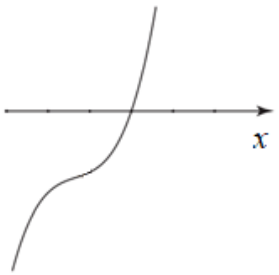


Figure 5

Now, the first question is: Given $a_3, a_2, a_1,$ and a_0, how do we distinguish cases of the above five categories?

Recall that $f'(x)$ is the first derivative of $f(x)$, and therefore, $f'(x)$ has the format $b_2x^2 + b_1x + b_0$. Then, we have $b_2 = 3a_3$, $b_1 = 2a_2$, and $b_0 = a_1$. The discriminant of $f'(x)$ is $b_1^2 - 4b_2b_0$, denoted as Δ. (The general form of quadratic function and its discriminant are well-known as $ax^2 + bx + c$ and $b^2 - 4ac$, respectively, in the literature.) The value of Δ is either greater than zero, equal to zero, or less than zero.

When $\Delta > 0$, there exist two values, say x_1 and x_2, such that $f'(x_1) = 0$, $f'(x_2) = 0$, and $x_1 < x_2$. These two values are the stationary points in the curve $y = f(x)$. The values of x_1 and x_2 can be found by using the quadratic formula. Precisely,

$$x_1 = \frac{-b_1 - \sqrt{\Delta}}{2b_2} \text{ and } x_2 = \frac{-b_1 + \sqrt{\Delta}}{2b_2}.$$

We check their corresponding y-values in the curve. Let $y_1 = f(x_1)$ and $y_2 = f(x_2)$. If y_1 and y_2 are both positive or both negative, then there is only one solution for $f(x) = 0$. This is Category 1. If one of y_1 and y_2 is zero, then there are two solutions for $f(x) = 0$. This is Category 2. If y_1 and y_2 are on opposite sides, (that is, one is positive and the other one is negative,) then there are three solutions for $f(x) = 0$. This is Category 3.

The case $\Delta = 0$ is referred to Category 4, and the case $\Delta < 0$ is referred to Category 5. There is only one solution for $f(x) = 0$ in these two categories.

In short, we can determine the number of solutions by the value of Δ, and the values of y_1 and y_2, for $\Delta > 0$.

The number of solutions is the first requirement in the objective. For fulfilling the objective, we also need to answer the second question: how do we find a solution?

Where are the solutions? We know that at least one solution exists between r and s, where $r < s$, $f(r) < 0$, and $f(s) > 0$.

Formally, since the curve is a continuous function, for any r and s, where $r < s$, if $f(r) < 0$ and $f(s) > 0$, then there exists a number t such that $r < t < s$ and $f(t) = 0$.

How to find r and s? Assign s to 1, and whenever $f(s) \le 0$, assign s to $2s$. Assign r to -1, and whenever $f(r) \ge 0$, assign r to $2r$. (For your interest, this method is called extended binary search.)

Continuing with r and s, we find the number t by using bisection method. We check the midpoint m of r and s. Precisely, $m = \frac{r+s}{2}$. If $f(m) > 0$, assign s to m and repeat the process with r and s; if $f(m) < 0$, assign r to m and repeat the process with r and s; if $f(m) = 0$, assign t to m. The solution is t.

This completes the theoretical background of the assignment.

The pseudocode below will include the above theoretical idea, as well as the practical techniques for handling the precision errors.

D. Pseudocode

```
Using appropriate data types, declare a3, a2, a1, a0, b2,
b1, b0, Delta, x1, x2, y1, y2, r, s, m, t, num_solution,
stop;

Ask the user politely to input a3, a2, a1, a0;

If a3 is no greater than zero,
    Output "Invalid input";
    Terminate;

b2 = (3)(a3);
b1 = (2)(a2);
b0 = a1;

Delta = (b1)^2 - (4)(b2)(b0);

If Delta is no greater than 0
    num_solution = 1,

Otherwise,
    x1 = [-b1 - square_root(Delta)] / [(2)(b2)];
    x2 = [-b1 + square_root(Delta)] / [(2)(b2)];
    y1 = (a3)(x1)^3 + (a2)(x1)^2 + (a1)(x1) + a0;
    y2 = (a3)(x2)^3 + (a2)(x2)^2 + (a1)(x2) + a0;

    If y1 and y2 are both positive
        num_solution = 1
    If y1 and y2 are both negative
        num_solution = 1
    If one of y1 and y2 is 0
        num_solution = 2
    If both y1 and y2 are 0
        Output "Precision error";
        Terminate;
    If y1 is positive and y2 is negative
        num_solution = 3
    If y1 is negative and y2 is positive
        num_solution = 3

Output num_solution;

If num_solution is 1

    s = 1;
    As long as (a3)s^3 + (a2)s^2 + (a1)(s) + a0 ≤ 0
        s = (2)(s);

    r = -1;
    As long as (a3)r^3 + (a2)r^2 + (a1)(r) + a0 ≥ 0
        r = (2)(r);
```

```
    stop = false;
    As long as stop is false
        m = the midpoint of r and s;
        Declare y_m with suitable data type;
        y_m = (a3)m^3 + (a2)m^2 + (a1)(m) + a0;
        If y_m is 0 or |s-r|<0.000001
            t = m;
            Output t;
            stop = true;
        Otherwise, If y_m is greater than 0
            s = m;
        Otherwise, If y_m is less than 0
            r = m;

Terminate;
// end of pseudocode
```

E. Sample Output

Sample test cases (inputs and corresponding outputs) are provided as below:

Test Case	Input	Output
1	**0 0 0 0**	**Invalid Input**
2	**3 0 -1 -1**	**1** **0.851383**
3	**1 -5 8 -4**	**2**
4	**1 -4 5 -1.9**	**3**
5	**4 0 1 0**	**1** **0.000000**

F. Requirement

You **MUST** use the corresponding **pseudocode** given in Section D. The function **`setprecision(int n)`** and **`pow(double x, int y)`** are allowed for all formulas and pseudocodes. Only **`<iostream>`**, **`<cmath>`**, and **`<iomanip>`** are allowed for the implementation. No function in other libraries is allowed.

MORE INSTRUCTIONS

1. **Assignment's Instructions:**

   a. This is **a group assignment**.
   b. You are recommended to read this assignment several times before coding.
   c. Cover the lecture topics from **Week 1 till Week 5, introduction till iteration**.
   d. Please take note that **your submitted assignment must be within the course scopes**. Any submission which is **OUT OF THE TOPICS** (such as Class, Linked List, Pointer,…) will be awarded with **ZERO MARK**.
   e. Please take note that your program should be developed according to the pseudocode given. Else, **ZERO mark** will be awarded.

2. **Report Submission**

   - **ZIP** the **SOFTCOPY** of your ***.cpp file, *.exe file and** the **REPORT (.pdf file) together and SUBMIT the .zip file** to WBLE before the due date, which is posted on WBLE.
   - If you do not know what is zip file, please refer to google or Wikipedia (https://en.wikipedia.org/wiki/ZIP_(file_format)) or 7zip (https://www.7-zip.org/).
   - The **.exe** file is very **IMPORTANT** and must be included into the zip file.
   - Remember to rename all the files (*.zip, *.cpp, *.exe, *.pdf) to your group number, E.g. Group30.exe. **NOTE: ONLY ON**E member will submit the files.
   - If you do not know what ***.exe file** is, please refer to practical 1.
   - The report is not expected to use more than 15 pages. Single line spacing, font size 11 and choose the font type accordingly. Exceeding pages will not be reviewed.
   - The **REPORT** should **consist** of the following items:

	Item	Description	Tick
i	Cover page	Your name, student ID, programme name (e.g., Bachelor of Computer Science), course code and name (to make sure we receive the correct assignment), group number and email address. **(*A sample is provided at the last page)**	☐
ii	Objective	Copy and paste from Section A.	☐
iii	Pseudocode	Copy and paste from Section D.	☐
iv	Flowchart	Draw a flowchart of your program. It must align with your pseudocode.	☐
v	Test cases	Screenshot of your program output. Please use 3 to 5 different inputs. Each test case should address a particular feature.	☐
vi	Source code	Source code, which is exactly the same as the softcopy submitted to WBLE	☐

3. **Suggest Grading Scheme**

   a) Correctness of the program according to the given pseudocode. It will mainly depend on thorough testing designed by the markers.
   b) Report structure: completeness, and quality of presentation of each item.
   c) Documentation of codes (how easy to understand your codes); However, avoid excessive comments, such as commenting every line of your code.
   d) Total marks for Assignment 1 are 40 marks (program 25 + report 15) and contributes 10% to the final grade;

e) The marking scheme shown below is just a **GUIDELINE** to develop the complete program as well as the report;
f) The marks maybe adjusted according to various circumstances;
g) The penalties will be complied as described below:

Suggested Marks		
	Details	**Suggested Grading Range**
A.	**Program**	
1.	Able to compile and able to execute	0 - 8
2.	Correct user interface	0 - 2
3.	Test cases	0 - 10
4.	Precision (exact output)	0 - 5
B.	**Report**	
1.	Pseudocode	0 - 2
2.	Flowchart	0 - 5
3.	Source code	0 - 3
4.	Screen shot	0 - 3
5.	Tidiness	0 - 2
	Subtotal marks (program)	25
	Subtotal marks (report)	15
	Grand Total	40
Penalties		
1	Late submission **OR** wrong submission (re-submission)	**10% deducted per day from total marks that earned**
2	**Out of topics** ✓ Class, Linked List, Pointer, Array, Enum, …. ✓ Other libraries that have not been mentioned in this guideline are prohibited. E.g. using #include <array> is prohibited	**-40**

*Grading ranges → Very Poor (0 - 15%), Poor (15% - 30%), Satisfactory (30% - 70%), Good (70% - 90%) and Excellent (90% - 100%)

UNIVERSITI TUNKU ABDUL RAHMAN

ACADEMIC YEAR 2022/2023

Wholly owned by UTAR Education Foundation
(Co. No. 578227-M)
DU012(A)

UCCD 1004 PROGRAMMING CONCEPTS AND PRACTICES

ASSIGNMENT 1

Group: XXX

	Student 1
Name:	
Student ID:	
Programme:	
Email:	

	Student 2
Name:	
Student ID:	
Programme:	
Email:	