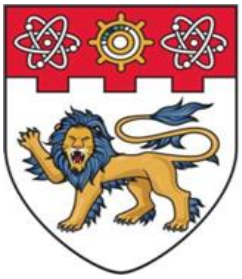


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

AY 2022/2023 Semester 2

MA4830 REALTIME SOFTWARE FOR MECHATRONIC SYSTEMS

CA1 REPORT

Tutorial Group: MA3

Group Members:

Name	Matric No.
Mohamed Raizee Bin Mohamed Ibrahim	U2022754H
Lin Yijuan	U2021676H
Wu Tong	U1923100L
Mohamad Hafiz Bin Mohamad Helmi	U2022464D

1. Introduction

This program is created to solve for the roots of a quadratic equation for users, by prompting users for the coefficients of the equation. Similar to a scientific calculator, this program takes the coefficients of the equation as inputs and returns the discriminant, type of roots and values of the roots as output. It also checks for validity of the inputs and should any input be inappropriate, the program will require a re-entry. This program is written in C and is executable on Windows OS and WSL: Ubuntu. The compiler used is GNU Compiler Collection (GCC). The program listing is available in **section 4.1 Appendix A.**

2. Methodology

In this section, we will explain how this program works to solve the equations and a flow chart will be used to illustrate the flow of the program.

2.1 Program implementation

- Firstly, the program will request the coefficient inputs from the terminal for the user to enter. The input accepts a string that must include digit(s) (e.g. **"12"– accept**).

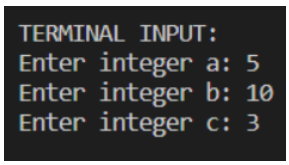


Figure 1

- If any of the string inputs from the user is identified to be not appropriate, the program will print the error message (e.g. "b must be an integer value!"). Thereafter, prompt the user for the inputs again. (More information for valid inputs under **section 2.2: Validity of Input**)

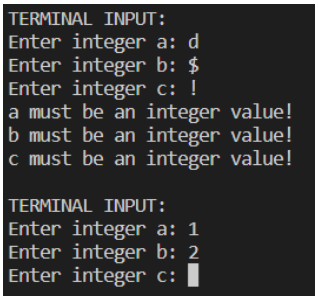


Figure 2

- Upon receiving valid input for all 3 coefficients, the program will perform arithmetic calculation to obtain the discriminant using the discriminant formula. Discriminant formula:

$$D = b^2 - 4ac$$

- From the discriminant obtained, the program will analyse and classify the type of roots into one of the following:
    - When D > 0, Type of roots: **2 real and different roots**

$$\text{Root 1: } \frac{-b+\sqrt{D}}{2a},$$

$$\text{Root 2: } \frac{-b-\sqrt{D}}{2a}$$

    - When D = 0, Type of roots: **1 real and same root**

$$\text{Root 1 = Root 2: } \frac{-b}{2a}$$

    - When D < 0, Type of roots: **Complex conjugate pair**

$$\text{Root 1: } \frac{-b}{2a} + \frac{\sqrt{|D|}}{2a}i,$$

$$\text{Root 2: } \frac{-b}{2a} - \frac{\sqrt{|D|}}{2a}i,$$

where $\frac{-b}{2a}$ is real part and $\pm\frac{\sqrt{|D|}}{2a}i$ is the imaginary part.

Subsequently, the program will solve for the roots according to the root type and display the inputs consisting of the coefficients and quadratic equation in a table, and the outputs consisting of discriminant, type of roots, and values of roots in another table, as shown in Figure 3 to 5 for the 3 different conditions (D > 0, D = 0 and D < 0):

```
INPUT DISPLAY:
-----------------------------------------------
|  a  |  b  |  c  |     Quadratic Equation     |
|-----|-----|-----|----------------------------|
|  5  |  10 |   3 |   5 (x^2)  +  10 (x) +   3 |
-----------------------------------------------

OUTPUT DISPLAY:
-------------------------------------------------------
| Discriminant |   Type of roots  | Root 1 |  Root 2  |
|--------------|------------------|--------|----------|
|     40.0     | REAL & DIFFERENT | -0.368 |  -1.632  |
-------------------------------------------------------
```

Figure 3: (For D>0)

```
INPUT DISPLAY:
-----------------------------------------------
|  a  |  b  |  c  |     Quadratic Equation     |
|-----|-----|-----|----------------------------|
|  1  |  6  |   9 |   1 (x^2)  +   6 (x) +   9 |
-----------------------------------------------

OUTPUT DISPLAY:
-------------------------------------------------------
| Discriminant |   Type of roots  | Root 1 | Root 2  |
|--------------|------------------|--------|---------|
|     0.0      |   REAL & EQUAL   | -3.000 | -3.000  |
-------------------------------------------------------
```

Figure 4: (For D=0)

```
INPUT DISPLAY:
-----------------------------------------------
|  a  |  b  |  c  |     Quadratic Equation     |
|-----|-----|-----|----------------------------|
|  2  | -6  |   5 |   2 (x^2)  -   6 (x) +   5 |
-----------------------------------------------

OUTPUT DISPLAY:
-------------------------------------------------------------------
| Discriminant |   Type of roots   |     Root 1     |     Root 2    |
|--------------|-------------------|----------------|---------------|
|     -4.0     | COMPLEX conjugate |  1.50 + 0.50i  |  1.50 - 0.50i |
-------------------------------------------------------------------
```

Figure 5: (For D<0)

- After solving a quadratic equation, the program will prompt the users whether they want to solve another quadratic equation. If users want to solve another quadratic equation, enter “y”. Else, enter any other keys to exit the program.

```
Do you want to solve another Quadratic Equation?
Type 'y' to continue or any other keys to exit:
```

Figure 6

2.2 Validity of Input

The program will check the validity of input values, and continue to prompt the user to re-enter all the coefficients until all the inputs are valid (as shown previously in Figure 2). In this section, we will elaborate on the correctness of the inputs with examples given.

- If there are missing input or symbols as an input, the program will not accept it, (e.g. **“-” , “$!” - reject**) as shown in Figure 7.

```
TERMINAL INPUT:
Enter integer a:
Enter integer b: -
Enter integer c: $!
a must be an integer value!
b must be an integer value!
c must be an integer value!
```

Figure 7

- Input with whitespaces before and after the digit(s) are still valid (e.g. **“ 15 ” -accept**) as shown in Figure 8, and the result will subsequently be displayed as shown in Figure 9.

```
TERMINAL INPUT:
Enter integer a:       15
Enter integer b:    20
Enter integer c:  33

Quadratic equation
15(x^2) + 20(x) + 33 = 0


DISPLAYING RESULT IN 2 SECONDS...
```

Figure 8

```
INPUT DISPLAY:
-----------------------------------------------
|  a  |  b  |  c  |     Quadratic Equation     |
|-----|-----|-----|----------------------------|
| 15  | 20  |  33 |  15 (x^2)  +  20 (x) +  33 |
-----------------------------------------------

OUTPUT DISPLAY:
-------------------------------------------------------------------------
| Discriminant |   Type of roots   |     Root 1     |     Root 2     |
|--------------|-------------------|----------------|----------------|
|    -1580.0   | COMPLEX conjugate | -0.67 + 1.32i  | -0.67 - 1.32i  |
-------------------------------------------------------------------------
```

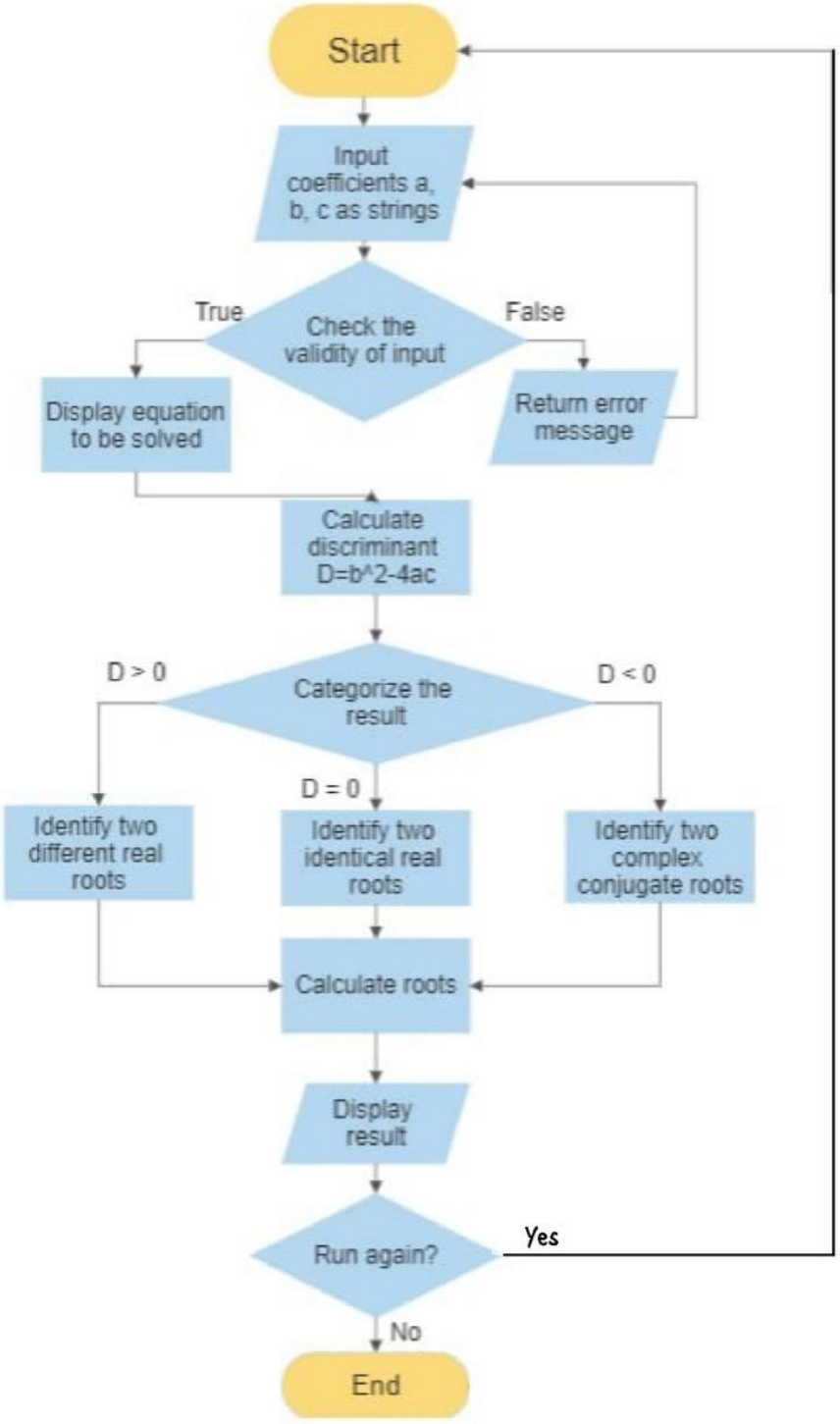
Figure 9

- If there is more than 1 digit, the program will not accept if the digits are separated by whitespace or symbols, including floating point. (e.g. **“4 5”, “23&1”, “1.50” - reject**) as shown in Figure 10.

```
TERMINAL INPUT:
Enter integer a: 4  5
Enter integer b: 23&1
Enter integer c: 1.50
a must be an integer value!
b must be an integer value!
c must be an integer value!
```

Figure 10

2.3 Flowchart of program



2.4 Program Novelty

- Able to accept input with whitespaces before and after the digits. This will allow the program to run even if the user unintentionally includes whitespaces at the start or at the end of their input.
- If input is incorrect, the program will prompt the user to re-enter the coefficients instead of terminating the program.
- The program will indicate which parameter(coefficient) is inputted in an incorrect format.
- The program displays the inputs and outputs neatly in a table format which is clear for the user to read the result.
- The output roots are in floats that display up to 3 decimal places for better accuracy.
- The program will prompt the user if they want to rerun the program to solve another quadratic equation or exit the program.
- Smooth and stable operation while using the program with no infinite looping or programming "hanging".

2.5 Program Limitations

- The only limitation in this program is that it does not accept floats as input:
    - Reason: Most quadratic equations have their coefficients in whole numbers.

3. Conclusion

With this program, users can solve quadratic equations easily within seconds by just providing the 3 coefficients input, instead of manually calculating which is time consuming and tends to have human error at times. This program is fast and accurate in solving the quadratic equations for users. Instructions on how to run the code can be found under **section 4.2 Appendix B: Instructions to run code from terminal.**

4. Appendix

4.1 Appendix A: Program listing/Source Code (Windows OS)

```
#include <stdio.h>  // for standard input output
#include <stdlib.h> // to access system to clear screen
#include <math.h>   // to utilise math functions
#include <ctype.h>  // for fgets()
#include <string.h> // to use strlen
#include <stdbool.h> // to use true false
#include <time.h>   // for time delay


#define BUFFER_SIZE 4096

bool check_int(char *string, int *integer);
void input_coeff(int* a, int* b, int* c);
void input_display(int a, int b, int c);
void clrscr();
void delay(int number_of_seconds);

int main()
{
    int a; // cant be zero otherwise it will be a linear eqn
    int b;
    int c;
    float x1, x2 = 0;  // The two roots
    float D, denom;    // D = Discriminant, denom = Denominator
    char rerun; // to check if user wants to rerun the program

    do{
        input_coeff(&a, &b, &c);
        printf("\n");
        printf("Quadratic equation\n");
        printf("%d(x^2) + %d(x) + %d = 0\n", a, b, c);
        printf("\n\n");

        // delay by 2 sec before clearing terminal
        printf("DISPLAYING RESULT IN 2 SECONDS...");
        delay(2);
        clrscr();

        D = b*b - 4*a*c;
        denom = 2 * a;

        printf("INPUT DISPLAY:\n");
        input_display(a, b, c);

        printf("OUTPUT DISPLAY:\n");


        if (D > 0)
        {
            x1 = (-b + sqrt(D)) / denom;  // D and denom are float, dont
need to typecast
            x2 = (-b - sqrt(D)) / denom;
```

```
            printf("-------------------------------------------------------
-\n");
            printf("| Discriminant |   Type of roots   | Root 1 |  Root 2
|\n");
            printf("|--------------|-------------------|--------|---------
|\n");
            printf("|     %-9.1f| REAL & DIFFERENT | %-7.3f| %7.3f | \n",
D, x1, x2);
            printf("-------------------------------------------------------
-\n");
        }

        else if (D == 0)
        {
            x1 = x2 = (float) -b / (2*a);      // typecast required as
all variables here are in int

            printf("-------------------------------------------------------
\n");
            printf("| Discriminant |   Type of roots   | Root 1 | Root 2
|\n");
            printf("|--------------|-------------------|--------|--------
|\n");
            printf("|     %-9.1f|   REAL & EQUAL    | %-7.3f| %6.3f | \n",
D, x1, x2);
            printf("-------------------------------------------------------
\n");
        }

    else
    {
            float real, img;
            real = (float) -b / (2*a);
            img = (float) sqrt(-D) / (2*a);

            printf("-------------------------------------------------------
----------------\n");
            printf("| Discriminant |   Type of roots   |     Root 1     |
Root 2    |\n");
            printf("|--------------|-------------------|----------------
|---------------|\n");
            printf("|    %-9.1f | COMPLEX conjugate | %5.2f + %4.2fi  |
%5.2f - %4.2fi | \n", D, real, img, real, img);
            printf("-------------------------------------------------------
----------------\n");
        }

    printf("\n");
    printf("Do you want to solve another Quadratic Equation?\nType 'y' to
continue or any other keys to exit: ");
    scanf("%c",&rerun);
    getchar(); //  newline buffer
    clrscr();   // reset terminal
    }while(rerun == 'y' || rerun == 'Y');

    // If user terminates program //
```

```
    clrscr();
    printf("Thank you for using our Quadratic Equation Solver Program!");
    printf("\n");
    return 0;
}

//////////////////// Function 1: If string format is able to return
integer, returns true ////////////////////
bool check_int(char *string, int *integer)
{
    /* ***This program only works for INT_MIN <= integer <= INT_MAX*** */
    int i = 0;

    while (isspace(string[i]))
    {
        i++; // as long it is whitespace, go to next character in the
string
    }

    int length = strlen(string);

    if (length == i) return false; // edge case where the whole string is
just whitespace

    char integer_buffer[BUFFER_SIZE]; // create an array of numeric
characters to track the index of each digit characters include '-'
negative symbol
    int integer_chars = 0;  // create index for valid integers character

    if (string[i] == '-')
    {
        integer_buffer[integer_chars] = '-';  // set '-' at index 0
        integer_chars++;                      // moves to next index
        i++;                                 // moves to the next char in
the string

        if (!isdigit(string[i])) return false; // if next char after '-'
is not int, return false
    }

    while (i < length && !isspace(string[i]))
    {
        if (!isdigit(string[i])) return false;  // if after '-' is a non
numeric character, returns false

        else
        {
            integer_buffer[integer_chars] = string[i]; // store the index
of next digit char from string i
            integer_chars++;
            i++;
        }
    }
    integer_buffer[integer_chars] = '\0';       // terminate the string
of numeric characters
```

```
    while (isspace(string[i])) i++;  // loop through every whitespace
after the numeric characters if theres any
    if (string[i] != '\0') return false; // return false if after those
whitespaces, there are still other characters: improper format

    *integer = atoi(integer_buffer); // convert the strings of numeric
characters into int and store into the dereferenced pointer value integer
    return true;
}

//////////////////// Function 2: Retrieve input coefficients of the
quadratic equations from user ////////////////////
void input_coeff(int* a, int* b, int* c)
{
    bool valid_integer_a = true;
    bool valid_integer_b = true;
    bool valid_integer_c = true;

    do
    {
        char buffer[BUFFER_SIZE];

        printf("\n");
        printf("TERMINAL INPUT:\n");

        printf("Enter integer a: ");
        fgets(buffer, BUFFER_SIZE, stdin);
        valid_integer_a = check_int(buffer, a);

        printf("Enter integer b: ");
        fgets(buffer, BUFFER_SIZE, stdin);
        valid_integer_b = check_int(buffer, b);

        printf("Enter integer c: ");
        fgets(buffer, BUFFER_SIZE, stdin);
        valid_integer_c = check_int(buffer, c);

        if (!valid_integer_a)
        {
            printf("a must be an integer value!\n");
        }

        if (!valid_integer_b)
        {
            printf("b must be an integer value!\n");
        }

        if (!valid_integer_c)
        {
            printf("c must be an integer value!\n");
        }
    } while (!valid_integer_a || !valid_integer_b || !valid_integer_c);
// checks if all a,b,c are valid integers
}

// Function 3: To display the input table //
void input_display(int a, int b, int c)
```

```
{
    printf("-------------------------------------------------\n");
    printf("|  a  |  b  |  c  |     Quadratic Equation     |\n");
    printf("|-----|-----|-----|----------------------------|\n");

    // conditional statement to check for the signs display of quadratic
equation
    if(b > 0 && c > 0)
    {
        printf("| %-3d | %-3d | %3d | %3d (x^2)  + %3d (x) + %3d |\n", a,
b, c, a, b, c);
    }

    else if(b < 0 && c > 0)
    {
        int bb = fabs(b); // returning absolute value of b
        printf("| %-3d | %-3d | %3d | %3d (x^2)  - %3d (x) + %3d |\n", a,
b, c, a, bb, c);
    }

    else if(b > 0 && c < 0)
    {
        int cc = fabs(c);
        printf("| %-3d | %-3d | %3d | %3d (x^2)  + %3d (x) - %3d |\n", a,
b, c, a, b, cc);
    }

    else
    {
        int bb = fabs(b);
        int cc = fabs(c);
        printf("| %-3d | %-3d | %3d | %3d (x^2)  - %3d (x) - %3d |\n", a,
b, c, a, bb, cc);
    }

    printf("-------------------------------------------------\n");
    printf("\n");

}

// Function 4:  To clear terminal //
void clrscr()
{
    system("@cls||clear"); // cls for windows command prompt, clear is
for unix based bash
}

// Function 5: to delay by indicated number of seconds //
void delay(int number_of_seconds)
{
    // Converting time into milli_seconds
    int milli_seconds = 1000 * number_of_seconds;

    // Storing start time
    clock_t start_time = clock();

    // looping till required time is not achieved
```

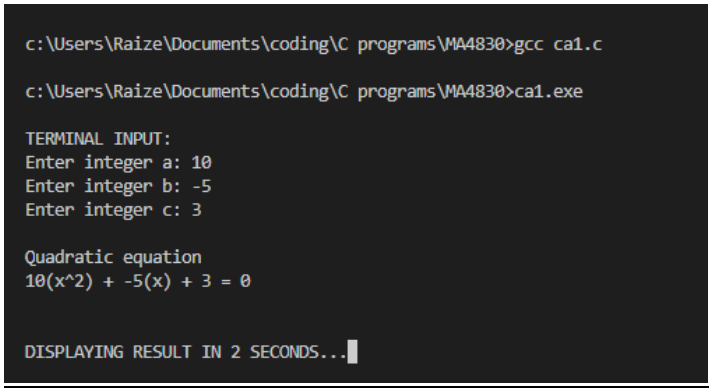
```
    while (clock() < start_time + milli_seconds)
        ;
}
```

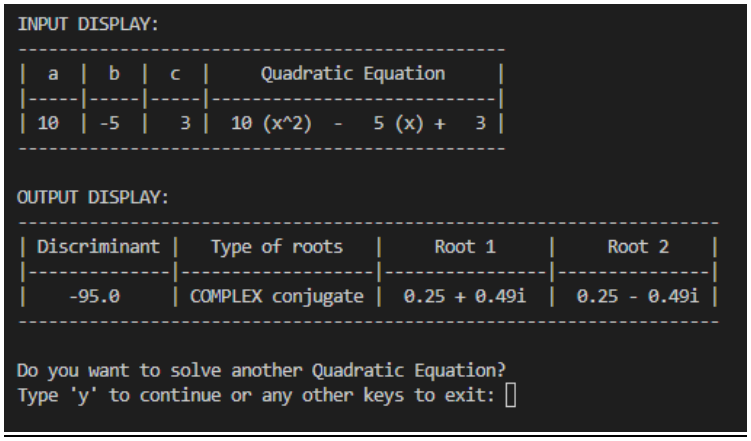
4.2 Appendix B: Instructions to run code from terminal. (Using gcc compiler)

Windows OS

Follow the steps below:

1. open command prompt
2. cd to working directory containing the source code .c file.
3. Type *gcc {filename}.c*
4. Type *{filename}.exe*





Windows Subsystem for Linux: Ubuntu

Modify the following lines of codes on text editor or IDE before running on Ubuntu:

```
7    // #include <time.h>   // for time delay
8    #include <unistd.h>
```

```
36          // delay(2);
37          sleep(2);
```

Then follow the steps below:

1. open ubuntu terminal
2. cd to working directory containing the source code .c file.
3. Type *gcc {filename}.c -o {execution_filename} -lm*
4. Type *./{execution_filename}*

```
root@MECHA:/home/raizee/coding/c/MA4830# gcc ca1.c -o ca1.exe -lm
root@MECHA:/home/raizee/coding/c/MA4830# ./ca1.exe

  TERMINAL INPUT:
  Enter integer a: 2
  Enter integer b: -6
  Enter integer c: 5
```

```
INPUT DISPLAY:
-----------------------------------------------
|  a  |  b  |  c  |     Quadratic Equation     |
|-----|-----|-----|----------------------------|
| 2   | -6  |   5 |   2 (x^2)  -   6 (x) +   5 |
-----------------------------------------------

OUTPUT DISPLAY:
-------------------------------------------------------------------------
| Discriminant |   Type of roots   |     Root 1     |     Root 2     |
|--------------|-------------------|----------------|----------------|
|    -4.0      | COMPLEX conjugate |  1.50 + 0.50i  |  1.50 - 0.50i  |
-------------------------------------------------------------------------

Do you want to solve another Quadratic Equation?
Type 'y' to continue or any other keys to exit:
```