ME-172
Computer Programming Language Sessional
Assignment No. 2

Abu Sayeed Roni, ID: 1710065
Dept: ME, Section: B-2

March 2, 2020

Problem 1

Write a C program to find the smallest of 3 integers taken as input using nested if-else statement.

Solution:

The code below could be written a little more succintly if we're allowed to use logical operators and if-else-if chaining. Or if the problem would've asked to find the smallest of, say, 100 integers, we would've definitely used a loop to iterate over the array of integers keeping track of the current smallest integer. But as the problem states that we are to use *nested if-else* statement, here is the code.

```
/**
 * Finds the smallest of three integers.
 */


#include <stdio.h>


int main(void)
{
    // Take user input for three integers.
    int a, b, c;
    printf("%s\n", "Enter three integers:");
    scanf("%i %i %i", &a, &b, &c);

    int smallest;

    // Compare three integers and find the smallest one.
    if (a < b)
    {
        if (a < c)
        {
            smallest = a;
        }
```

```
            else
            {
                smallest = c;
            }
        }
        else
        {
            if (b < c)
            {
                smallest = b;
            }
            else
            {
                smallest = c;
            }
        }

        // Show output.
        printf("Smallest integer: %i\n", smallest);
    }
```

Saving the file with name `smallest.c`. Then Compiling the source file with the following command:

```
$ clang -o smallest smallest.c
```

An executable binary file with name `smallest` will be created if everything goes right.

Output

Running the executable with the following command:

```
$ ./smallest
Enter three integers:
6
3
9
Smallest integer: 3
```

Problem 2

Write a C program to find the roots of a quadratic equation $ax^2 + bx + c = 0$, that will take coefficients a, b, and c as input and find the roots as output. Use nested if-else statement.

Solution:

Roots of a quadratic equation $ax^2+bx+c = 0$, where the constants $a, b, c \in \mathbb{R}$ and $a \neq 0$ is given by the following formula:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

However, the nature of the roots are going to depend on the discriminant $b^2 - 4ac$. Here are the possible cases.

- If $b^2 - 4ac > 0$ then the roots of the quadratic equation $ax^2 + bx + c = 0$ are real and unequal.
- If $b^2 - 4ac = 0$ then the roots of the quadratic equation $ax^2 + bx + c = 0$ are real and equal.
- If $b^2 - 4ac < 0$ then the roots of the quadratic equation $ax^2 + bx + c = 0$ are complex conjugate.
- If $b^2 - 4ac > 0$ and is a perfect square then the roots of the quadratic equation $ax^2 + bx + c = 0$ are rational and unequal

Now that we know what the roots are going to be like before we ever solved the equation for them, we can compute the *third case* from above a bit differently than the other three cases.

If $b^2 - 4ac < 0$, then $-(b^2 - 4ac) = 4ac - b^2$ is a *positive* quantity. Which will help us to resolve the formula into real and imaginary parts like the following.

$$\begin{aligned} x &= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\ \Rightarrow x &= \frac{-b}{2a} \pm \frac{\sqrt{4ac - b^2}}{2a} i \end{aligned}$$

All we need to do now is translate this resolved formula into C and we are good to go!

```
/**
 * Finds the roots of a quadratic equation.
 */


#include <stdio.h>
#include <math.h>
#include <stdbool.h>


int main(void)
{
    // Take user input for coefficients a, b, and c.
    double a, b, c;
    printf("%s", "a = ");
    scanf("%lf", &a);
    printf("%s", "b = ");
    scanf("%lf", &b);
    printf("%s", "c = ");
    scanf("%lf", &c);

    // Calculate
    double discriminant = pow(b, 2) - 4 * a * c;
    bool hasRealRoots = discriminant < 0 ? false : true;
    double root1, root2;
    double realPart, imginaryPart;
    if (hasRealRoots)
    {
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
    }
    else
    {
        realPart = -b / (2 * a);
```

```
            // Discriminant, discriminant is negative here
            // as hasRealRoots boolean variable is evaluated
            // false in the if clause. So, negating discriminant
            // will return a non-negative argument for sqrt function.

            // What we are actually doing is pulling out the
            // imaginary i off the expression so that we can just
            // append it later when we will be showing output.
            imginaryPart = sqrt(-discriminant) / (2 * a);
        }

        // Show output
        printf("Roots are: \n");
        if (hasRealRoots)
        {
            printf("%.2lf, %.2lf\n", root1, root2);
        }
        else
        {
            // As the roots are complex conjugate, we write the
            // real and imaginary parts separately which we
            // calculated into two separate variables beforehand.
            printf("%.2lf+%.2lf%c, %.2lf-%.2lf%c\n",
             realPart, imginaryPart, ’i’, realPart, imginaryPart, ’i’ );
        }
    }
```

Saving the file with name `quadratic.c`. Then compiling the source file with the following command:

```
$ clang -o quadratic quadratic.c -lm
```

An executable binary file with name `quadratic` will be created if everything goes right.

Output

Running the executable using the following command:

```
$ ./quadratic
a = 1
b = 2
c = 3
Roots are:
-1.00+1.41i, -1.00-1.41i
```

The output is as expected which is equal to $-1 + \sqrt{2}i$ and $-1 - \sqrt{2}i$ repectively, except our output is rounded to two decimal places.

NOTE: All the programs are written in *Linux* environment. The executable binary files don't have an extension like .exe, .dmg, or .app beacuse in Linux whether a program is executable or not is determined by the permissions on the file, not the extension. And LLVM's *Clang* is used for the compilation of above source files.