APS 105 — Computer Fundamentals

Lab #3: Decisions and Simple Loops
Winter 2022

You must use `examify.ca` to electronically submit your solution by 11:59 pm on **Saturday, February 5, 2022**.

In Visual Studio Code, place your solution for each part in a separate directory, so that Visual Studio Code will not attempt to compile multiple files and link them together.

Objective

The goal of this laboratory is to practice the material on decision makings as well as loops. You will be writing two separate and fun C programs as provided.

Preparation

Read through this entire document carefully, and do the work to create the programs that are described in Part 1 and 2 as provided in the following pages, and try to make them work.
Start and follow the suggested steps below:

1. Write your pseudocode in a plain language describing and planning the steps in solving the problem at hand.
2. Follow up your plan by writing your code on paper, i.e. stay away from typing on your computer at this time.
3. Try to review your code and identify any mistakes that you can identify.
4. Use your computer, type and run your code.
5. Run the test cases provided to ensure proper functionality of your code.

Notes:

- In the sample output examples that follow:
  - The text that would be entered by the program user is displayed using a **bold** font.
  - The text **<enter>** stands for the user pressing the `enter` key on the keyboard. On some keyboards, the `enter` key may be labeled `return`.
- When you execute your programs, the user's text will not be displayed in bold and **<enter>** is not going to show.
- Throughout this lab, there is a single space after the colon (:) in each output line.

Part 1 — Substance Test

Table 1 below shows the normal boiling points of several substances. Write a C program that prompts the user for the observed boiling point of a substance in degree Celsius ($^\circ C$) (as an integer), and identifies the substance if the observed boiling point is within $t^\circ C$ of the expected boiling point, where t is a threshold defined by the user. If the data input is not within $t^\circ C$ of any of the boiling points in the table, the program should output the message: `Substance unknown.`

Here is a sample output from an execution of the program:

Table 1: Substances and their normal boiling points.

Substance	**Normal boiling point**
Water	100
Mercury	357
Copper	1187
Silver	2193
Gold	2660

```
Enter the threshold in Celsius: 10<enter>
Enter the observed boiling point in Celsius: 2668<enter>
The substance you tested is: Gold
```

Here is another example:

```
Enter the threshold in Celsius: 30<enter>
Enter the observed boiling point in Celsius: 1156<enter>
Substance unknown.
```

Note: you may assume that the entered values by the user are valid integers within the range allowed by the `int` type. The test cases will only use threshold values sufficiently small to yield unique answers.

Part 2 — The Triangle

Write a complete C program that uses the `*` character to draw a triangle of a given number of rows. The program first prompts the user to enter the number of rows in the triangle. Your program may assume that the input is a valid integer from 1 to 20 (inclusive).

Here are some sample outputs from the execution of the program. The output of your program should match the sample output.

Sample output 1:

```
Enter the number of rows in the triangle: 1<enter>
*
```

Sample output 2:

```
Enter the number of rows in the triangle: 2<enter>
 *
***
```

Sample output 3:

```
Enter the number of rows in the triangle: 3<enter>
  *
 * *
*****
```

Sample output 4:

```
    Enter the number of rows in the triangle: 10<enter>
             *
            * *
           *   *
          *     *
         *       *
        *         *
       *           *
      *             *
     *               *
    *******************
```

Hint: You may find it helpful to draw the required output on a piece of graph paper before writing your program.

Part 3 — Change

In this part, you are asked to write a program that makes change for amounts less than one dollar. User input should be a positive integer value less than 100, which represents an amount of cash, in cents. The program should prompt the user for a such a positive integer value, and should then print the original amount of cash, together with a set of coins (quarters, dimes, nickels, cents) that could make up that amount. The program should produce change containing the **minimum** number of coins required for the given amount. The output should be in a natural, non-stilted form. For example, input of 58 should produce the following output:

```
58 cents: 2 quarters, 1 nickel, and 3 cents.
```

rather than something like

```
58 cents: 2 quarters, 0 dimes, 1 nickels, 3 cents.
```

Invalid input (0, negative values or positive values of 100 or more) should be rejected by your program as shown in the examples below. Your program should repeatedly prompt the user for the next amount, until an invalid amount has been entered. When an invalid amount has been entered by the user, the program will terminate.

Sample Run

Shown below is a sample run of the program. The values `10`, `16`, `20`, ... are entered by the user in each sample run. Given the same user input, your output should match the following output *exactly*, including all punctuation. Any variation from this will result in a loss of marks with some test cases.

```
Please give an amount in cents less than 100: 10
10 cents: 1 dime.
Please give an amount in cents less than 100: 16
16 cents: 1 dime, 1 nickel, and 1 cent.
Please give an amount in cents less than 100: 20
20 cents: 2 dimes.
Please give an amount in cents less than 100: 28
```

```
28 cents: 1 quarter and 3 cents.
Please give an amount in cents less than 100: 30
30 cents: 1 quarter and 1 nickel.
Please give an amount in cents less than 100: 36
36 cents: 1 quarter, 1 dime, and 1 cent.
Please give an amount in cents less than 100: 67
67 cents: 2 quarters, 1 dime, 1 nickel, and 2 cents.
Please give an amount in cents less than 100: 75
75 cents: 3 quarters.
Please give an amount in cents less than 100: 99
99 cents: 3 quarters, 2 dimes, and 4 cents.
Please give an amount in cents less than 100: -3
-3 cents: invalid amount.
```

Marking

This lab will be marked out of 10. 1 mark for coding style (introduced and discussed at length in Plenary Lecture 2), and 3 marks for each part in the lab. The coding style will be marked in your lab section by your lab TA. Here are the basic requirements for your code to earn the coding style mark:

- Use meaningful identifiers for the variables in your C program. Use either snake case or camel case, but be consistent throughout your entire C program.
- Use named constants when needed, rather than literal constants.
- Use consistent indentation throughout your entire C program.
- Use comments at the beginning of your program before the `main()` function. You may also add comments throughout your program as necessary, but they are not strictly enforced.
- Follow coding formatting guidelines for nested `if` statements, as discussed in our lectures.

The automarker may use multiple test cases for each part, and if this is the case, the marks for this part will be evenly split among the test cases. The test cases used for marking may or may not be the same as the test cases that are made available to you. The deadline of **(11:59 p.m. on Saturday, February 5, 2022)** is strictly enforced, so avoid last minute submissions.

Stay Safe and Good Luck!