APS 105 — Computer Fundamentals

Lab 4: Repetition and Decision Making

The goal of this laboratory is to practice the material on mainly loops as well as decision making. You will be writing two separate and fun C programs. The first part is to determine what coins (cents, nickels, dimes and quarters) can be used to make up a given amount of money. The other program requires the "drawing" of a triangle of various sizes depending on user's input.

Preparation

This lab has two parts. Read through this entire document carefully, and do the work to create the programs that are described in Part 1 and 2 below, and try to make them work. You can do this on your own home computer if you have succeeded in downloading and getting Codelite to work, as described in Lab 0. You can also do this work in the ECF labs prior to your lab period.

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— Making Change

In a file called **Lab4Part1.c**, write a complete C program that makes change for amounts less than one dollar. That is, you will determine the number and amount of coins that can make up a specific number of cents. Valid input to the program should be a positive integer value less than 100, which represents an amount of money, in cents. The program should prompt the user for a positive integer value, and in response to user input, should print the original amount of cash, together with a set of coins (quarters, dimes, nickels, cents) that 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 form, in that whenever there is one coin, the coin type should be singular, otherwise, the coin type should be in its plural form. If the number of coins for a particular coin type is 0, the type should not be shown. For example, if user input is 58 cents, the output must look like this:

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

The following shows an example of an incorrect output:

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

The program should repeatedly prompt the user for additional user input, until an invalid input is entered, after which the program should stop. An invalid input would be a negative number, one that is greater than 99, or less than 1. It can be assumed that the user always enters integer values when prompted.

An example run of the program is shown below. The values 10, 16, 20, . . . are entered by the user in the example run. Given the same user input, your output should match the following output *exactly*, including all punctuation, and all wording (including the plural form if needed). Any variation from this will result in a loss of marks.

```
Please give an amount in cents less than 100: 10<enter>
10 cents: 1 dime.
Please give an amount in cents less than 100: 16<enter>
16 cents: 1 dime, 1 nickel, 1 cent.
Please give an amount in cents less than 100: 20<enter>
20 cents: 2 dimes.
Please give an amount in cents less than 100: 28<enter>
28 cents: 1 quarter, 3 cents.
Please give an amount in cents less than 100: 30<enter>
30 cents: 1 quarter, 1 nickel.
Please give an amount in cents less than 100: 36<enter>
36 cents: 1 quarter, 1 dime, 1 cent.
Please give an amount in cents less than 100: 67<enter>
67 cents: 2 quarters, 1 dime, 1 nickel, 2 cents.
Please give an amount in cents less than 100: 75<enter>
75 cents: 3 quarters.
Please give an amount in cents less than 100: 99<enter>
99 cents: 3 quarters, 2 dimes, 4 cents.
Please give an amount in cents less than 100: 0<enter>
Goodbye
```

Part 2 – The Triangle

In a file called **Lab4Part2.c**, 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.

Grading by TA and Submitting Your Program for Auto-Marking
There are a total of 10 marks available in this lab, marked in two different ways:

1. By your TA, for 4 marks out of 10. Once you are ready, show your program to your TA so that we can mark your program for style, and to ask you a few questions to test your understanding of what is happening. Programs with good style are:
    - Clear comments that describe what is happening in the program.
    - Good choices for variable names that indicate their purpose. Please adopt the naming convention where if you have a variable that is described by multiple words, user lower case for the first letter of the first word, and Upper case for all subsequent words e.g. inputCode.
    - Properly indented code.
    - Proper use of named constants, rather than putting constants (such as 125) directly into the code.

    The TA will also ask you some questions to be sure that you understand the underlying concepts being exercised in this lab.

2. By an auto-marking program for 6 marks out of 10. You must submit all of your program files through the ECF computers for marking. We will use a software program to compile and run your program, and test it with different inputs. Long before you submit your program for marking, you should run the exercise program that compiles and runs your program and gives it sample inputs, and checks that the outputs are correct. You should run the following command:

/share/copy/aps105s/lab4/exercise

within the directory that contains both your solution programs. This program will look for the files comprising Lab4 in your directory, compile them, and run them on some of the test cases that will be used to mark your program automatically later. If there is anything wrong, the exercise program will report this to you, so read its output carefully, and fix the errors that it reports.

IMPORTANT: YOU WILL HAVE TO COPY ALL OF YOUR FILES TO BE IN THE SAME FOLDER/DIRECTORY WHERE YOU WILL RUN THE SUBMIT COMMAND.

3. Once you have determined that your program is as correct as you can make it, then you must submit your program for auto-marking. This must be done by the end of your lab period as that is the due time. To do so, go into the directory containing your solution files and type the following command:

    /share/copy/aps105s/lab4/submit

    This command will re-run the exercise program to check that everything looks fine. If it finds a problem, it will ask you if you are sure that you want to submit. Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked. All files to be submitted as solutions to the lab must be within the same directory, and the submission script must be run also from that directory via the command line. The exercise program (and the marker program that you will run after the final deadline) will be looking for the exact letters as described in the output in this handout, including the capitalization. When you test your program using the exercise program, you will see that it is expecting the output to be exactly this, so you will have to use it to see if you have this output correct.

Important Note: You must submit your lab by 11:59 p.m. after the end of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.

You can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:

/share/copy/aps105s/lab4/viewsubmitted

This command will download into the directory you run it in, a copy of all of the files that have been submitted. If you already have files of that same name in your directory, these files will be renamed with a number added to the end of the filename.

After the Final Deadline Obtaining Automark

Briefly after all lab sections have finished you will be able to run the automarker to determine the automarked fraction of your grade on the code you have submitted. To do so, run the following command:

/share/copy/aps105s/lab4/marker00

This command will compile and run your code, and test it with all of the test cases used to determine the automark grade. You will be able to see those test cases output and what went right or wrong.

Good Luck!