APS 105 — Computer Fundamentals

Lab 3: Decision Making and Repetition

Winter 2018

The goal of this laboratory is to practice the material on decision making and loops. You are to write two separate 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 is to build a simple calculator.

Instructions:

- Similar to previous labs, your solution will be marked by your TA during your scheduled lab period for programming style and your understanding, **and** should also be submitted electronically by the end of your scheduled lab period.

Preparation:

- Read through this entire document carefully, and do the work to create the programs that are described below. You are encouraged to:
  - Read the class bulletin board on Piazza, to see if others had similar problems. Ask a question if you do not find anything helpful. Do not ask for, or ever give a full or partial solution to the lab assignment.
  - Ask for assistance from your lab/tutorial TAs.
  - Attend and ask questions during the plenary lecture.

All labs in this course are due at the end of your scheduled lab period. This means that you **must** have both been graded by your TA, and you **must** have gone through the submission process described in this lab. The same process will be used in every one of the upcoming labs. Lab 3 is due on February 2 (Friday) or February 5 (Monday), depending on your lab section.

Part 1 — Making Change

In a file called **Lab3Part1.c**, write a complete C program that makes change for amounts less than one dollar. That is to 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. The program assumes 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 loss of marks.

In the sample output examples that follow, the user input appears after the colon character (':') and is followed by the **enter** key. Note that there is a single space after the colon (:) in each output line.

```
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, 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, 3 cents.
Please give an amount in cents less than 100: 30
30 cents: 1 quarter, 1 nickel.
Please give an amount in cents less than 100: 36
36 cents: 1 quarter, 1 dime, 1 cent.
Please give an amount in cents less than 100: 67
67 cents: 2 quarters, 1 dime, 1 nickel, 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, 4 cents.
Please give an amount in cents less than 100: 0
Goodbye.
```

Part 2 — Character-based Calculator

Write a program that acts like a calculator and asks for two double numbers for calculations based on one of the operations of addition, subtraction, multiplication, or division. The program asks the user to input a single character - one of **a, s, m** or **d** which mean that the two numbers should be added, subtracted, multiplied or divided, respectively. Your program should also obey the following specifications:

1. The specific calculation should be: firstNumber **operator** secondNumber, where **operator** is one of +, -, *, /, and where firstNumber is the first number the user inputs, and secondNumber is the second.
2. If the user's second Number is zero, and the division operation is requested, then your program should output the following: "Error, trying to divide by zero"
3. If the user inputs a character other than one of the four given, then your program should output: "Error, unknown calculation command given"
4. The outputs should display two decimal places using the proper specifier.
5. Your C program must go in a file named **`Lab3Part2.c`**.

Below are several sample outputs from an execution of the program, which are separated by lines such as:

`___Run 1____`

In the sample output examples that follow, the user input appears after the colon character (':') and is followed by the **enter** key. Note that there is a single space after the colon (:) in each output line.

```
__________Run 1________
Enter first number: 4.0
Enter second number: 5.0
Enter calculation command (one of a, s, m, or d): a
Sum of 4.00 and 5.00 is 9.00
__________Run 2________
Enter first number: 55.6
Enter second number: 75.8
Enter calculation command (one of a, s, m, or d): s
Difference of 55.60 from 75.80 is -20.20
__________Run 3________
Enter first number: 3.4
Enter second number: 2.8
Enter calculation command (one of a, s, m, or d): m
Product of 3.40 and 2.80 is 9.52
__________Run 4________

Enter first number: 90.0
Enter second number: 41.0
Enter calculation command (one of a, s, m, or d): d
Division of 90.00 by 41.00 is 2.20
__________Run 5________
Enter first number: 37.9
Enter second number: 0.0
Enter calculation command (one of a, s, m, or d): d
Error, trying to divide by zero
__________Run 6________
Enter first number: 3.2
Enter second number: 1.8
Enter calculation command (one of a, s, m, or d): p
Error, unknown calculation command given
```

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 have been described in class, but briefly, they are:

    a. Clear comments that describe what is happening in the program.

    b. Good choices for variable names that indicate their purpose. Please adopt the following naming convention illustrated above — if have you 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. **blendMode**.

    c. Properly indented code that has appropriate spacing between lines for readability.

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 should run the following command:

   ```
   /share/copy/aps105s/lab3/exercise
   ```

   Within the directory that contains your solution programs.

   This program will look for the files **Lab3Part1.c** and **Lab3Part2.c** in your directory, compile them, and run them on some test cases. 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.

   A key part of what you are to learn in this lab is the use of this program - most software programs give this kind of report. Read through the output of the **exercise** program and see if it is happy with everything, or if it is reporting an error. If there is an error, it will say so, and then it is up to you to fix what is wrong and try again. You'll need to do this for every subsequent lab.

   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 **Lab3Part1.c** and **Lab3Part2.c** file (and make sure this is the right place!) and type the following command:

   ```
   /share/copy/aps105s/lab3/submit
   ```

   This command will re-run the exercise program to check that everything looks OK. 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.

   Important Note: You must submit your lab by 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/lab3/viewsubmitted
   ```

After the Final Deadline — Obtaining Automark

After all lab sections have finished, a short time later, 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/lab3/marker
```

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!