Dalhousie University
CSCI 2132 — Software Development
Winter 2017
Assignment 3

Distributed Wednesday, February 1 2016.
Due 5:00PM, Wednesday, February 15 2016.

Instructions:

1. The difficulty rating of this assignment is *bronze*. Please read the course web page for more information about assignment difficulty rating, late policy (no late assignments are accepted) and grace periods before you start.

2. Each question in this assignment requires you to create one or more regular files on bluenose. Use the exact names (case-sensitive) as specified by each question.

3. C programs will be marked for correctness, design/efficiency, and style/documentation. Please refer to the following web page for guidelines on style and documentation for short C programs (which applies to this assignment):

   `http://web.cs.dal.ca/~mhe/csci2132/assignments.htm`

   You will lose a lot of marks if you do not follow the guidelines on style and documentation.

4. Create a directory named `a3` that contains the following files (these are the files that assignment questions ask you to create): `a3q1.txt`, `a3q2.txt`, and `a3q3.c`. Submit this directory electronically using the command `submit`. The instructions of using `submit` can be found at:

   `http://web.cs.dal.ca/~mhe/csci2132/assignments.htm`

5. Do NOT submit hard copies of your work.

Questions:

1. [8 marks] Give concise answers to the two questions below. Use either `emacs` or `vi` to type your answers in one single plain ASCII file, with `a3q1.txt` as its name. Each line of this file should have at most 80 characters.

   (i) [4 marks] Each job under the job control facility of the Bash shell has a job ID and a PID. Are these two identifiers the same identifier for each job? A simple yes or no answer will be sufficient.

(ii) [4 marks] Suppose that you have written and compiled a C program, and the executable file is named `computation`, which is stored in the current working directory. Suppose that this program typically requires more than an hour to run, so you would like to make it run in the background, and also direct its output to a file named `answer` and its error messages to a file named `exception`. Write down a command that could perform this required task. You can assume that your current working directory contains the file `computation`.

2. [10 marks] For either of the following pairs of `scanf` format strings, indicate whether or not the two strings are equivalent. If they are, explain why. If not, show how they can be distinguished by providing one input. Explain why this input distinguishes them.

   Use either `emacs` or `vi` to type your answers in one single plain ASCII file, with `a3q2.txt` as its name. Each line of this file should have at most 80 characters.

   (i) [5 marks] `"%d"` versus `" %d"`

   (ii) [5 marks] `"%f,%f"` versus `"%f, %f"`

   Use either `emacs` or `vi` to type your answers in one single plain ASCII file, with `a3q2.txt` as its name. Each line of this file should have at most 80 characters.

3. [12 marks] For this question, you will write a C program. Since you are supposed to be able to work on it after one or two lectures on C, this is an easy question. However, it is important to follow the problem specification, especially the input and output format, as we perform automatic testing to test the correctness of your program.

   As this is the first programming question, you can find a draft of its marking scheme at: `http://web.cs.dal.ca/~mhe/csci2132/assignments/a3q3.pdf`

   Make sure to read the marking scheme.

   This question asks you to write a program to help you make changes using as few coins as possible. That is, given a cash amount in cents, your program will compute how many of each coin you need to make changes, and the total number of coins should be minimized. You are only allowed to use toonies (200 cents), loonies (100 cents), quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent). For example, for 372 cents, you need 1 toonie, 1 loonie, 2 quarters, 2 dimes and 2 cents.

   Mathematically, this problem can be modeled as follows: Given an integer x, where $0 \leq x < 500$, you want to express

   $$x = 200a + 100b + 25c + 10d + 5e + f$$

   where a, b, c, d, e, f are nonnegative integers, and $a + b + c + d + e + f$ is minimized.

You are required to use the following greedy algorithm to solve this problem: You first choose as many toonies as possible, and the for the remaining amount, choose as many loonies as possible, and so on. This is how we get the solution to the above example.

You might wonder why this algorithm is correct; see additional notes at the end of this question for more details.

General Requirements: Use `a3q3.c` as the name of your C source code file.

We will use the following command on bluenose to compile your program:

```
gcc -o coins a3q3.c
```

Your program should NOT print anything to prompt for user input. It reads a single integer from stdin. You can assume that this integer is a value from (and including) 0 to (but excluding) 500.

Your program then prints the result as six integers, which, from the first to the last, are the numbers of toonies, loonies, quarters, dimes, nickels and pennies required. Separate any two numbers next to each other using one single space character. Do not output any space characters after the last integer. Output a newline symbol at the end so that the result will be printed on a separate line.

Therefore, when you run your program by entering `./coins`, the program will wait for your input. If you enter `372`, the program will output `1 1 2 2 0 2`

Testing: To test your program automatically, we will use it as a UNIX filter.

For the example above, we will test it using the following command in the directory containing the executable program:

```
echo 372 | ./coins
```

The output should be:

```
1 1 2 2 0 2
```

To help you make sure that the output format of your program is exactly what this questions asks for, several files are provided in the following folder on bluenose to show how exactly your program will be automatically tested:

```
/users/faculty/prof2132/public/a3test/
```

In this folder, open the file `a3test` to see the commands used to test the program with two test cases. The output files, generated using output redirection, are also given.

To check whether the output of your program on any test case is correct, redirect your output into a file, and use the UNIX utility `diff` (learned in Lab 3) to compare your output file with the output files in the above folder, to see whether they are identical.

Since these output files are given, we will **apply a penalty to any program that does not strictly meet the requirement on output format**. This includes the case in which your output has extra spaces, or has a missing newline at the end.

We will use different test cases to test your program thoroughly. Therefore, construct a number of test cases to test your program thoroughly before submitting it.

Useful tips: Since we have not learned how to use a debugger, a useful tip for debugging is to add some `printf` functions to print the values of variables during execution, and this will give you insights on what might have gone wrong. Make sure to remove these statements before you submit.

Additional notes: The greedy algorithm you are required to implement is correct for the Canadian system of coinage. It does not always work for any arbitrary coinage system. For example, consider a system of coins of 12 cents, 5 cents and 1 cent. Then for 15 cents, the greedy algorithm will use 1 coin of 12 cents and 3 coins of 1 cent each. This requires 4 coins in total. However, a better solution is to use 3 coins of 5 cents each.

Why is this greedy algorithm correct for the Canadian system of coinage? You will be able to come up with a rigorous proof after you learn Algorithm I, which will teach you how to design greedy algorithms, i.e. algorithms that make the locally optimal choice at each stage.