Dalhousie University
CSCI 2132 — Software Development
Winter 2018
Assignment 3

Distributed Wednesday, January 31 2018.
Due 9:00PM, Wednesday, February 14 2018.

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_a.txt`, `a3q1_b.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. [9 marks] This question is on UNIX shells and processes.

   (i) [5 marks] Develop a *single* command line that prints a message of the following format:

   ```
   The current time is Wed Jan 31 18:15:57 AST 2018
   ```

In this message, the part after "is" must be the current date and time when you run this command line.

Use `script` to record your most successful execution of the command line in a file named `a3q1_a.txt`.

(ii) [4 marks] How to make a foreground process run in the background? Give a concise answer. Use either `emacs` or `vi` to type your answers in one single plain ASCII file, with `a3q1_b.txt` as its name. Each line of this file should have at most 80 characters.

2. [9 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 `a3q2.txt` as its name. Each line of this file should have at most 80 characters.

   (i) [4 marks] Is the string `2ndday` a legal identifier in the C programming language? Justify your answer.

   (ii) [5 marks] The follow two statements, when used in a C program, can print two lines of poems from the Lord of the Rings:

   ```
   printf("All that is gold does not glitter,\n");
   printf("Not all those who wander are lost.\n");
   ```

   Write a single statement that is equivalent to these two statements. That is, your single statement should call the function `printf` once to print the same lines of text. Since this statement will be longer than 80 characters, for this question only, you are allowed to break it into two lines when typing your solution (though in most cases lines can not be broken arbitrarily in real C programs).

   Note that this is for practice only. In real programs, readability is often more important.

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 introductory lecture 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 compute the digits of the octal number that represents the permission of a file. More precisely, your program will read nine binary digits that represent the permission of a file, and then output three octal digits that represent the same permission.

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 perm a3q3.c
```

Your program should NOT print anything to prompt for user input. It reads nine integers from stdin, and each integer is either a 0 or a 1. You can assume that the input is always correct. That is, no error handling is required for this question.

Your program then prints the result as three integers, which, from the first to the last, are octal digits representing the permission of user, group and other, respectively. 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 `./perm`, the program will wait for your input. If you enter `1 1 0 1 0 0 1 0 0`, the program will output `6 4 4`

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 1 1 0 1 0 0 1 0 0 | ./perm
```

The output should be:

```
6 4 4
```

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 against 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. **Previously there were students who did not use diff to compare, and ended up losing marks because of incorrect output format.**

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.