CISC 3320 HW Assignment – 2 (10 pts)

This assignment tries to combine the management of processes and threads (Ch3&4). Specifically, you are asked to launch a new process from within another process, and the new process will run multiple threads concurrently to do some statistics analysis of a file with all English words

Requirements

1. Use either C/C++ or Java to write two programs: `LaunchProcess` and `AnalyzeWords`
2. In `LaunchProcess`
   - Allow user to provide a file name as command-line argument
   - The file `english_words.txt` will be provided to you
   - Your program will
     - launch a new process that runs your second program: `AnalyzeWords`
     - wait for the second process to finish
     - print its exit value
     - calculate and display how many milliseconds it takes to finish
3. In `AnalyzeWords`
   - Allows user to provide the words file name as command-line argument
   - Launch 3 threads to do the following analyses and report their result
     - the average length of a word
     - the longest length of all words
     - the most frequently used letter in all words
4. Sample command lines and their output:

```
$./AnalyzeWords english_words.txt                    ← as a native app
Average length = ????
Longest word's length = ???
Most frequent letter = ?

$java LaunchProcess AnalyzeWords english_words.txt   ← as a Java app
Average length = ????
Longest word's length = ???
Most frequent letter = ?
Exit value is: ?
Time to finish all tasks: ??? ms
```

Hints

While the implementation in C will be straightforward and relevant methods have been covered in slides, implementation in Java is likely novel to many of you even if you're familiar with Java `Threads`.

For doing `LaunchProcess`, you need to develop some understanding of Java's `Process` class and its API. The following website gives good examples: https://www.geeksforgeeks.org/java-lang-process-class-java/. There is something unique to Java Process, i.e., the subprocess does not have its own console and standard I/O has to be redirected to the parent process. For example, your second program's output must be piped into the first program's input to be displayed (please refer to the aforementioned website for an example of the `getInputStream`() method)

For doing `AnalyzeWords`, how to launch Java threads is covered in the Ch4 slides. While you can choose to launch and manage individual `Thread`s, a more convenient approach is to use the thread pool afforded by the `ExecutorService` (see slides #31-32 in Ch4). The thread pool allows the use of `Callable` (besides `Runnable`) so that your thread can return a value. This way you can display result in the main thread. It's recommended that you finish `AnalyzeWords` first, and make sure that it outputs reasonable result before connecting with the `LaunchProcess` program.

What to turn in

- Please submit (**via email**) the following:
  - your source code in separate files (don't list your source code in PDFs)
  - output of the result (you could use a screen capture, no videos)
- Please do not copy code, whether from classmates or from the Internet.