Lab 1: Mastery of previous stuff

14 January 2021

This week's lab will work a bit differently from the later labs in the semester: it will be a number of essentially separate problems to work on, to make sure that everyone's on the same page and to give you a chance to brush up on the stuff you haven't done for a month and a half (or more).

Logging in, setting up, copying files

This section is not meant as a comprehensive tutorial on Linux usage, (consult your lab notes from 160 for more detail), but rather as a reminder of some commands you haven't used in a while. They should be enough to get you started, but don't hesitate to ask questions if you're stuck.

First, log in to `cs.longwood.edu` using PuTTY (or ssh, if you're on Mac or Linux).

Set up this semester's directory, and today's work.

```
mkdir 162
cd 162
mkdir lab1
cd lab1
```

(On future logins, you won't have to `mkdir` again, at least not for `162`; and from your home directory you can change directly to `162/lab1` if you like.)

If you type `ls` at this point, the directory should be empty; if you type `pwd` ("print working directory") it should show you you're inside `/home/yourlogin/162/lab1`.

Now you'll need to copy the starter files for this lab. They're in `/home/shared/162/lab1/`, and you should copy all the files there into your working directory for this lab. Since you're currently *in* that directory, you can use the shortcut "`.`" (a single dot), which always refers to the current working directory:

```
cp /home/shared/162/lab1/* .
```

Type `ls` again to check that the files are there.

The actual work before us

The core work of the week is just to solve these fourteen problems that you previewed last night. Each has a description below and some setup and testing information in the files you've just copied into your directory. Read the readme file! Among other things, it gives instructions on how to use the tests I've provided for you.

About the tests: For each problem, one or two tests are provided. If you write code that passes them, that likely means you have the format correct, but it's no guarantee that your code is correct: the provided test cases are not comprehensive and do not even try to test logical edge cases. You'll have to test your own code to be sure that it works!

The functions

The handin for each of these should be one single `.cpp` file. It may contain multiple functions (including, potentially, the answers to other problems) but only the function named in the problem (and helpers it calls, if any) will be graded. The file `labfunctions.h` contains the headers that your functions must match to be considered correct. Each function has a provided tester with just one or two cases; feel free to add your own test cases to the tester files (which will not be handed in).

a. `lastVowel` finds and returns the last vowel in a given string. Vowels are AEIOU (upper or lowercase). Returns `'!'` if there are no vowels in the phrase.

   Hand in as `lab1a` using

   ```
   handin cmsc162 lab1a yourfilename.cpp
   ```

b. `numVowelsIncludingY` counts the number of vowels in a given string. In addition to AEIOU, Y is here considered a vowel, unless it is followed by another vowel.

   Hand in as `lab1b` using

   ```
   handin cmsc162 lab1b yourfilename.cpp
   ```

c. `lastNameFirst` converts a given full name to be in the format last name, followed by a comma and space, followed by the rest of the name. Here, "last name" is determined to be everything after the first space in the full name.

   Hand in as `lab1c` using

   ```
   handin cmsc162 lab1c yourfilename.cpp
   ```

d. `getInitials` returns the "initials" of a given phrase (the series of "first" letters in each word, where words are separated by a single space).

   Hand in as `lab1d` using

   ```
   handin cmsc162 lab1d yourfilename.cpp
   ```

e. `bookPageEntry` creates a "page entry" for a book with given title and given author. The entry is a single string of length 70, with the name of the book left justified and the author's name right justified. Between are dots separated by single spaces; none of the dots should be immediately adjacent to either the title or the author. If an extra space is needed to make the width add up to 70, it should be immediately before the author's name (making a double space in that spot).

   Hand in as `lab1e` using

   ```
   handin cmsc162 lab1e yourfilename.cpp
   ```

f. `isRightTriangle` determines whether three given integers can be lengths of sides of a right triangle. That is, if they are, in any order, a Pythagorean Triple satisfying $a^2 + b^2 = c^2$.

   Hand in as `lab1f` using

   ```
   handin cmsc162 lab1f yourfilename.cpp
   ```

g. `nearestFive` takes a given whole number of cents and rounds to the nearest nickel (so 42 rounds down to 40, 53 rounds up to 55, and so on).

   Hand in as `lab1g` using

   ```
   handin cmsc162 lab1g yourfilename.cpp
   ```

h. `piFromEuler` computes the kth approximation of π according to Euler's series

$$6 \cdot \left(\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots + \frac{1}{k^2} \right) \to \pi^2$$

The series approaches π^2 as k gets bigger, but your code should be computing π itself.

Hand in as `lab1h` using

```
handin cmsc162 lab1h yourfilename.cpp
```

i. `sumOfDigits` computes the sum of all of the digits in the given positive integer.

Hand in as `lab1i` using

```
handin cmsc162 lab1i yourfilename.cpp
```

The programs

The handin for each of these should be one single `.cpp` file. It may contain multiple functions but should have exactly one `main` function; it should not rely on any other files in the directory. Each program has provided one or two test inputs and corresponding expected output files; feel free to write your own additional test files (which will not be handed in).

If you test by hand (running the program without using the test files), you'll need to know that Ctrl-D is the indication that keyboard input is complete (corresponding to the end of the file in the test-file-based input).

j. `numWordsUntilSTOP` reads the input, word by word, until it sees the sentinel value "`STOP`", then prints the number of words it read and a newline. Words are separated by whitespace. The count of words should not include the occurrence of "`STOP`" (nor words that occur after that, if any).

Hand in as `lab1j` using

```
handin cmsc162 lab1j yourfilename.cpp
```

k. `acrostic` reads every line of the input and prints the first letter of each, followed by a single newline character.

   Hand in as `lab1k` using

   ```
   handin cmsc162 lab1k yourfilename.cpp
   ```

l. `checkPrimes` reads positive integers until it finds a zero; for each, it prints a single line containing either "`PRIME`" or "`NOT PRIME`" as appropriate.

   Hand in as `lab1l` using

   ```
   handin cmsc162 lab1l yourfilename.cpp
   ```

m. `gradesSummary` reads grades until it runs out of input, then prints one line with "`Pass: `" and the number of grades that were passing (at least 55), and another line with "`Fail: `" and the number that were failing (less than 55).

   Hand in as `lab1m` using

   ```
   handin cmsc162 lab1m yourfilename.cpp
   ```

n. `multTable` reads two positive integers, representing a width and then a height, and prints a multiplication table with 1 at the upper left. The width tells how many numbers should be printed across the table; each number should be printed in two columns and be separated from the next number by an additional space. (The lines should not end in a space.) The height tells how many rows the table should have.

   Hand in as `lab1n` using

   ```
   handin cmsc162 lab1n yourfilename.cpp
   ```

Grading rubric

Labs will generally be graded out of ten points, and each week, I plan to provide a rubric for how I plan to award points. This gives you some idea of where to focus your attention if you run short on time, and can act as a checklist to make sure you haven't forgotten anything.

This week:

RUBRIC

General

1 Present in lab

1 ... with pseudocode written down for at least a few of them

½ Handing in at least one correct by 4pm tomorrow (Friday)

½ Getting at least 10/14 correct by the Wednesday "due date"

Problems

½ Each

Handing in

This week is funny for several reasons; usually there will be a single handin for the week (not fourteen separate ones), and a fixed due date, (usually 4pm the following Wednesday) and that information will be noted/reminded in a section at the end of the lab handout.

This week, the due date is indeed 4pm next Wednesday (the 20th), but it is a soft deadline: there is a half point in the rubric tied to it, but other than that it's a rolling deadline. Hand each part in when you think it's done, and I'll check them when I get a chance. There's no formal time limit on finishing this particular lab.

However, *not* finishing the lab is not an option (or at least, mostly finishing it). As mentioned in the syllabus, this is a mastery evaluation, and you have to master the skills from 160. **If you don't at least *eventually* get 10/14 correct, your grade in this course maxes out at D+.** This is intended as a strong motivator to get your basics in order (and preferably soon, in the first week or two of the term); but it's not meant to be punitive or exclusive. If you're stuck on problems, talk to me about them and I'll help you through them (as will the tutor(s) once the tutoring is set up for the semester). So, persevere, don't panic, you'll be fine.