Lab 1: Brushing off the cobwebs
Preview

19 January 2017

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.

Before you come to lab, you should read each description and write pseudocode for *at least* four of them, on paper, and bring them with you. (I'll check. They don't have to be perfect, but they should be a good-faith best effort.) You *can* do more, and if you aren't sure how to proceed on some of them, that would be a *great* thing to bring up in lab tomorrow.

For some of these the task will be "write a function" and for others it will be "write a complete program", but that shouldn't matter at this stage. I will provide some starter code and some tests you'll be able to run, so don't get too ahead of yourself on typing anything in.

The problems

They might not be in this order in the lab handout tomorrow; use their names in your notes. Some are definitely easier than others, but they are not ordered by difficulty.

1. `numWordsUntilSTOP` reads the input, word by word, until it sees the sentinel value "`STOP`", then prints the number of words it read. Words are separated by whitespace.
2. `lastVowel` finds the last vowel in a phrase. Vowels are AEIOU (upper or lowercase). Produces `'!'` if there are no vowels in the phrase.
3. `numVowelsIncludingY` counts the number of vowels in an input. In addition to AEIOU, Y is here considered a vowel, unless it is followed by another vowel.
4. `lastNameFirst` converts a person's name to be in the format last name followed by a comma followed by the rest of the name. Here, "last name" is determined to be everything after the first space.

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

6. `acrostic` reads every line of the input and prints the first letter of each.

7. `bookPageEntry` creates a "page entry" for a book. 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. The result may contain a double space before the author's name.

8. `isRightTriangle` determines whether three given numbers can be lengths of sides of a right triangle. That is, if they are a Pythagorean Triple satisfying $a^2 + b^2 = c^2$. Note that your code must work regardless of the order of the values.

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

10. `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.

11. `gradesSummary` reads grades until it runs out of input, then prints the number of grades that were passing (at least 55) and the number that were failing (less than 55).

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

13. `checkPrimes` reads numbers until it finds a zero; for each number, it prints either "`PRIME`" or "`NOT PRIME`".

14. `multTable` prints a multiplication table of a specified width and height. Numbers less than 10 should still be printed in two columns so that everything lines up nicely.