CSE 130 Midterm 1 Instructions

1. Please sit at the machine to which you have been assigned. You can find the machine number on a label on the lid of each computer. **Be sure to sign the attendance sheet as well.**

2. Complete **ALL** of the problems on the pages that follow. The score for a perfect final exam is 100 points.

3. Partial credit may be assigned at the instructor's discretion. **Programs that do not compile will receive 0 points.**

4. The exam is open-book, open-notes, limited Internet. You will only have access to one Web site during the exam (the course Web page) which can be found at:

   http://www.cs.stonybrook.edu/~tashbook/spring2017/cse130/

5. **You may not use cell phones, tablets, e-readers, pagers, or any other electronic devices during the exam, except for USB flash drives, USB mice, and the computers that are installed in the exam room.** Please turn all prohibited devices **OFF** for the duration of the exam.

6. You **may not** communicate in any fashion (notes, texting, talking, etc.) with any other students during the exam. **Any suspected cheating will result in a grade of 0 for the exam and academic dishonesty charges.**

7. At the start of the exam, your instructor will tell you where to find the questions and the starting code for the exam problems. You will also be provided with instructions for submitting your completed work.

1. **How Many Habaneros? (25 points)**

   Four kinds of peppers grow in my garden — poblanos, jalapeños, habaneros, and serranos — all intermingled. Given a sequence of one or more characters (each character representing one type of pepper), complete the "peppers.c" program, which counts and prints the total number of habanero peppers there are in my garden. Each pepper is identified by its first letter (for example, an 's' in the input represents a serrano pepper). A capital letter represents a fully-grown pepper, and a lowercase letter represents a still-growing pepper; this doesn't affect the count, but in this problem it means that you need to count ***both*** uppercase and lowercase versions of 'h'. You may assume that the input sequence ***only*** contains 'p', 'j', 'h', and 's' characters (both uppercase and lowercase), and that it ends with a newline ('\n') character.

   Hint: You don't know how long the input sequence will be, so use a `while` loop with `getchar()` to process the user input until it is complete.

   Sample input: `hSPPjJHHsshpjs`

   Sample output: 4

2. **Largest Digit (25 points)**

   Complete the "largest.c" program, which reads in a positive, non-zero integer value from the keyboard. The program determines and prints out the largest digit in the input. For example, given the input 17452, the program should report that 7 is the largest digit.

   Hint: One way to solve this problem is to process the input from right-to-left, using modulo and division to repeatedly remove the rightmost digit until you reach 0.

3. **Base Conversion (25 points)**

   To convert a number from any arbitrary base N into base 10 (decimal), we multiply each digit, working from right to left, by an increasing power of N. The rightmost digit is multiplied by the base raised to the 0th power (which is just 1). The next-to-rightmost digit is multiplied by the base raised to the first power. The next digit is multiplied by the base squared, and so on until the last (leftmost) digit is reached. The base 10 value of the number is equal to the sum of these products.

   For example, consider the base 4 value 1322. We multiply the rightmost digit, 2, by 4^0, or 1, to get 2. The next digit, also 2, is multiplied by 4^1 to get 8. The digit after that, 3, is multiplied by 4^2 (which is 16) to get 48. Finally, the last/leftmost digit, 1, is multiplied by 4^3 (or 64) to get 64. The sum of 2, 8, 48, and 64 is 122 (in base 10).

Complete the "converter.c" program, which reads in two positive integers from the user: a number in some base between 2 and 10 inclusive, and the specific base in which that number is currently represented. The program should print out the base 10 (decimal) value of the first input.

Hint: Work from right-to-left using modulo and division, like you did for the previous problem.

4. **Give Me Five! (25 points)**

I love the number five. In fact, I only care about numbers that have at least one 5 among their digits. Complete the "givemefive.c" program, which takes two positive, non-zero integers as input: the starting and ending values of a range (you may assume that the first value is always less than the second value). The program prints out every number in the range (including the starting and ending values) that contains at least one 5 among its digits, and ***only*** those values. When it finishes, the program should ***also*** print out the total number of values that it printed out.

For example, given the input values 42 and 75, your program should produce output similar to the following:

45 50 51 52 53 54 55 56 57 58 59 65 75
13 value(s) printed.

Hint: I recommend creating a helper function that determines whether a given integer contains at least one 5 among its digits (modulo and division will be helpful here as well).