CSCI 1933 Lab 3

Recursion, Iteration and String I/O

Lab Rules

You may work individually or with *one* partner in lab. We suggest that you have a TA evaluate your progress on each milestone before you move onto the next. The labs are designed to be completable by the end of lab. If you are unable to complete all of the milestones by the end of lab, you have **until the last office hours on Monday** to get those checked off by **any** of the course TAs. We suggest you get your milestones checked off as soon as you complete them since Monday office hours tend to become crowded. If you worked with a partner, both of you must be present when getting checked off to receive credit. You will only receive credit for the milestones you have checked off by a TA. There is nothing to submit to Moodle for this lab.

Attendance

Per the syllabus, students with more than 3 unexcused lab absences over the course of the semester will automatically fail the course. The TAs will take attendance during lab; it is expected that students will be actively working on the lab material. *Your physical presence in lab is not sufficient to be marked as present for the purposes of attendance.*

1 Reversing a Number

In this section, you will reverse a given number. To do so, write a `Reverse` class with the following methods:

- `public static int recursiveReverse(int num)` – This will reverse `num` **recursively**
- `public static int iterativeReverse(int num)` – This will reverse `num` **iteratively**

You may assume that `num` ≥ 0. As always, write test cases for your methods. We have provided some test cases on the next page to get your started. We expect you to write **at least 3 test cases for each method**.

Caution: Integers that end with 0 will not properly reverse if you are doing this mathematically since leading zeros (after reversing) will not be retained. Your method does not need to handle reversing integers ending with 0's. However, your method should properly handle zeros that are embedded within an integer, e.g., 109.

Constraint: You cannot use any `reverse()` methods you did not write yourself.

Examples:

- `recursiveReverse(57)` will return 75.
- `recursiveReverse(19924)` will return 42991
- `recursiveReverse(9876543)` will return 3456789
- `recursiveReverse(4000002)` will return 2000004
- `recursiveReverse(111111)` will return 111111

The same test cases apply for the iterative counterparts

Milestone 1:
When you are done, explain the working of the two functions to a TA to get this section checked off. *If you were given a very large number to reverse, say a trillion, which approach would be better: recursive or iterative?* Discuss your answer with a TA. If you are stuck or have questions, do not hesitate to ask us!

2 Fibonacci Numbers

The Fibonacci numbers are named after the Italian mathematician, Leonardo of Pisa, also known as Fibonacci. Applications of the Fibonacci numbers include the Fibonacci search technique and in the analysis of the bounds of the Fibonacci heap data structure. The Fibonacci sequence begins with *fibonacci(0) = 0* and *fibonacci(1) = 1* as its respective first and second terms. After the first two elements, each subsequent element is equal to the sum of the previous two elements.

Write a `Fib` class with the following **recursive function**: `fibonacciRecursive(int n)` that takes in a number, n, and produces the n^{th} Fibonacci number. Write a main driver method to test your method. Use the `Scanner` class to prompt an integer input from the user.

Examples:

- `fibonacciRecursive(3)` = 2
- `fibonacciRecursive(5)` = 5
- `fibonacciRecursive(8)` = 21
- `fibonacciRecursive(10)` = 55

Constraint: In no shape or form may you use the closed form expression for the Fibonacci numbers.

Milestone 2:
Try to run your function with the following input arguments - 4, 6, 14, 20, 50, 60, 80, 100, 200. When you are done, explain the working of the function to a TA to get this section checked off. *What happens when you enter in the last three arguments?* Discuss your answer with a TA and try to come up with an approach to solve the problem. If you are stuck or have questions, do not hesitate to ask us!

3 Reverse Vowels of a String

In this section, you will take a string from the user as input from the command line. For taking input from the command line, you can use the `Scanner` class. Write a `ReverseVowels` class with the following method.

- `public static String reverseVowel(String s)` – This will reverse the vowels in the string.

Your function should not fail for `null` strings and should print *"Incorrect input"*. Write a main driver method to test your methods. We expect you to write **at least 3 test cases for each method**.

Hint: You might want to look into the `toCharArray()` method of the String class

Hint: You can have two pointers, called start and end, pointing at either ends of the string. If the characters at those positions are vowels, then swap and move the pointers to the middle! If the pointer points to a consonant, then move it without swapping.

Examples:

- `reverseVowel("hello")` will return *holle*.
- `reverseVowel("golden)` will return *geldon*.
- `reverseVowel("gophers")` will return *gephors*.

Milestone 3:
When you are done, explain the working to a TA to get this section checked off. If you are stuck or have questions, do not hesitate to ask us!