**CS1401 – Programming Assignment 8**

**Due: Monday November 9, 2015 by 8:00 a.m. for those of you taking the MW labs**

**Tuesday November 10, 2015 by 8:00 a.m. for those of you taking the TR labs**

**To be submitted via Piazza in the folder “lab8”**

**Welcome to Lab 8!**

|  |
| --- |
| This week, more and more on methods. This time, you will get to experience repetition as recursion and iteration (one implementation each). You will also practice on linking recursion to logic programming with prolog. This will allow you to gain more experience in concepts covered earlier in the semester (loops, arrays, strings) but now also in the context of recursion.  Finally, you will also have to show us that you know how to read code, how to trace it, and how to modify it.  We hope you have fun!  Ok, let’s get started! Here are the two activities you will be working on. |

** You should expect to work about 3 to 4 extra hours outside the lab session to complete this assignment.**

**It means that you need to make sure that Java works on your own computer, or that you go to open labs to work some more.**

**Extra time on labs includes completing the activities and taking the time to make sure that your submission is picture perfect!**

 **It’s all about repetition.**

Note: logic programming, as it is a different programming, might feel uncomfortable. Remember: be patient and think… and you will manage!

Repetition is everywhere in computer programs. Most of the time, the reason why we even “bother” to program is because the solution to our problem requires repetitions that we are not willing to carry out ourselves. As a result, understanding and practicing repetition is crucial to the success of a computer scientist.

In class, we have studied for and while loops. These allow us to implement repetition in what we call an iterative way (one set of instructions, followed by another similar set of instructions, etc.). We also saw that another way of implementing repetition is via recursion (in which a function calls itself).

In this lab, you will have to implement independent methods that involve repetition of set of instructions. For each of them, you will have to implement, in Java, an iterative and a recursive version. For two of them, you will have to implement them in Prolog as well.

**Activity 1. Factorial**

The factorial of a number is defined as follows:

Factorial(n) = n x (n-1) x (n-2) x … x 2 x 1, for all n >= 1

Factorial(0) = 1

You are expected to design a method that computes any factorial number:

* Your method should take as an input an integer n.
* Your method should return an integer that is the nth factorial number.

You need to produce two pseudo-codes and two Java methods:

* One iterative version, using a for loop, called IterFactorial;
* One recursive version, using recursion (no loop), called RecFactorial; as well as:
* One Prolog version of the recursive method you designed and implemented in Java, simply called factorial, which takes two parameters: n and the value of the factorial of n.

**Activity 2. Fibonacci**

Fibonacci numbers are defined as the following series:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …,

where every Fibonacci number is the sum of the previous two Fibonacci numbers:

Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2), for n >= 2

Fibonacci(1) = 0

Fibonacci(2) = 1

You are expected to design a method that computes any Fibonacci number:

* Your method should take as an input an integer n.
* Your method should return an integer that is the nth Fibonacci number.

You need to produce two pseudo-codes and two Java methods:

* One iterative version, using a for loop, called IterFibonacci;
* One recursive version, using recursion (no loop), called RecFibonacci.

Challenge question (20 extra points):

Run both IterFibonacci and RecFibonacci on input n = 50.

1. Observe and describe what happens.

2. Explain what happens, specifically referring to the way the methods are implemented and what exactly is being executed.

**Activity 3. Say-it-out-loud (SIOL) Sequence**

The say-it-out-loud sequence start with any number you want, say 1, and then goes as follows:

1, 11, 21, 1211, 111221, 312211, …

which is just the way you would say it: each number is the number before said out loud.

Let’s look at the sequence again:

* We started with 1
* This is one 1: we write 11
* These are two ones: we write 21
* This is one 2 and one 1: we write 1211
* This is one 1, one 2, and two ones: 111221
* etc.

You are expected to design a method that computes any SIOL number:

* Your method should take as an input two integers:
  + an integer start, which is the seed integer of the sequence; and
  + an integer n, which is the length of the sequence you have to generate.
* Your method should return an array containing all numbers in the sequence up to the nth number.

You need to produce one pseudo-code and one Java method:

* One iterative version, using a for loop, called IterSIOL.

Extra-points: using auxiliary methods as relevant will result in extra points (up to 20 points extra).

**Activity 4. Palindrome**

A word is a palindrome if it has the same spelling whether read from left to right or from right to left. For instance, kayak is a palindrome and racecar is a palindrome.

You are expected to design a method as follows:

* Your method should take as an input one String
* Your method should return true if the input String is a palindrome, false otherwise.

You need to produce two pseudo-codes and two Java methods:

* One iterative version, using a for loop, called IterPalindrome;
* One recursive version, using recursion (no loop), called RecPalindrome.

Notes: 1/ you are allowed to use the method substring() only for the recursive version; 2/ lower/upper case should not matter: as a result “KayAk” should also be considered a palindrome; 3/ you are allowed to use the method toLowerCase() in both versions.

**Activity 5. Selection sort**

Manipulating data is central to what you will be doing as a computer scientist. In class, we have studied variables and types, and recently arrays as a more advanced way of storing information. You will see that it is often necessary to sort information given to you in an array. There exist different sorting algorithms, among which selection sort, a very simple to grasp way of sorting data.

In this activity, you will have to:

1. Explain in your own words how this algorithm works.
2. Write pseudocode for it in your own words.
3. Understand the code provided to you and trace its execution on the following two arrays:
   1. {5,4,3,2,1}
   2. {10,1,7,2}
4. Write pseudocode for a recursive selection sort with the same input and output as the iterative selection sort we provided.
5. Implement the corresponding recursive selection sort.

Note: You might need to implement an auxiliary method for the recursive version of selection sort.

**What you have to turn in:**

* In a docx file, write the pseudocode of each of the above methods: remember that a good pseudocode can be handed out to a programmer without further explanation and the programmer should be able to translate it to code. This is the standard that you will be held to.
* In the java file that was provided to you, complete the description of the methods as described above and according to your pseudocode + include in the main method the lines of code that are necessary to execute each of the above methods.
* In the Prolog file that was provided to you, complete the description of the two methods you have to implement.

**Important Note:** If your code is not properly indented, it will not be graded (you will receive 0 for the coding part of your grade).

**Important notes:**

* Indent your code properly following guidelines available at: <http://www.oracle.com/technetwork/java/javase/documentation/codeconventions-136091.html>. Badly-indented code will be graded 0.
* Spend time working on your pseudocode as the amount of points you get for the pseudocode is bigger than the amount of points you get for your code (usually, close to a 60/40 ratio).
* Do not submit more than the files that are requested from you: one docx file, one Java file, and one Prolog file.

**That’s all! Looking forward to seeing you in lab!**