**CS1401 – Challenge Lab 1**

**Due: Monday November 30, 2015 by 11:59 p.m.**

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

**Challenge Lab 2!**

|  |
| --- |
| This lab is your second and last challenge lab. As for the first one, it is OPTIONAL and if you decide to do it and you are successful, you can obtain up to 100 points to be added to your total number of lab points (50 points per activity). This will definitely increase your final grade for labs!  Important: since this is an optional lab, you are expected to work on it outside your lab sessions, unless you are completely done with the current assigned lab.  We hope you enjoy being challenged once again! |

**This lab is not in replacement of the regularly assigned labs. Regularly assigned labs should be your priority and only once you are done, you should work on this challenge lab. You should expect to work about 3 to 4 hours to complete this assignment.**

**Activity 1/ Lab description. Sudoku checker.**

In this activity, you will design a method called **SudokuChecker** whose aim is to check if a given Sudoku board is properly filled, i.e., whether it satisfies the Sudoku rules (see Wikipedia, <https://en.wikipedia.org/wiki/Sudoku>, for more information about Sudoku if you are not fully familiar with it).

Method **SudokuChecker** takes the following input:

* A 9x9 2D array that contains integers from 1 to 9.

And it returns true if the Sudoku board is properly filled, false otherwise.

*Note: the 9x9 2D array passed as a parameter to your method should be acquired via reading a file that contains the numbers. This auxiliary method (which you have to implement as well) is called* ***readSudokuBoard****: it takes a file name as parameter and returns a 2D array of integers.*

**What you have to turn in:**

* A docx file in which you describe the pseudocode of the above two methods (SudokuChecker and readSudokuBoard), along with a set of test cases designed to challenge your implementation and ensure that it works properly (i.e., as expected).
* A **single** java file that contains the above two methods as well as a main file that allows to test your codes.

**Activity 2/ Let’s be logic!** In this activity, you will have to practice programming in Prolog, mostly so that you see that there are multiple ways to implement the solution to a problem.

**Activity 2.1.** Implement in prolog a predicate isEmpty that takes a list L. isEmpty(L) is true if L is empty, false otherwise.

*Hint: in prolog, you do not have to define what is false: you just do not mention it (a prolog file only contains true information).*

**Activity 2.2.** Implement in prolog a predicate isElement that takes a list L and an element E. isElement(E,L) returns true if E is an element of L, false otherwise.

*For instance: isElement(3,[1,2,3,4,5]) is true, but isElement(6,[1,2,3,4,5]) is not.*

**Activity 2.3.** Implement in prolog a predicate everyOther that takes two lists L1 and L2. everyOther(L1, L2) is true if L2 is the list made of every other element of L1, false otherwise.

*For instance: everyOther([1,2,3,4,5],[1,3,5]) is true, but everyOther([1,2,3,4,5],[1,2,3]) is not.*

**Activity 2.4.** Implement in prolog a predicate sudokuSolution that takes a Sudoku board as parameter and is true if this board satisfies the rules of Sudoku (numbers in the board are only numbers from 1 to 9, there are no duplicates in any row, column, 3x3 region) and false otherwise.

***Note:*** *you have implemented a similar method in Java in the first activity of this lab. Please make sure to go back to that code and compare the approaches. You will see that for a given problems and a given approach to solving this problem, codes (from one language to another) can be very different.*

**What you have to turn in:**

* The code (in your prolog file named challenge2.pl) of each of the methods (predicates) as requested. In addition, for Activity 2.4., please tell us what you think about this code, compared to the Java code you wrote for the same problem just before.
* The tests you run on each of these predicates to see if your implementation is correct, and why (in the docx file).

**Important notes:**

* Indent your code properly following guidelines available at: <http://www.oracle.com/technetwork/java/javase/documentation/codeconventions-136091.html>. Badly indented code may be returned without being graded.
* Spend time working on your pseudocode as pseudocode weighs 50% of the grade on this lab.
* Do not write your methods inside the main method. Each method has to be written where instructed in the code provided to you. Failing to do so puts you at risk of losing 30% off your lab grade.
* Do not submit more than the files that are requested from you: one docx file and one java file.

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