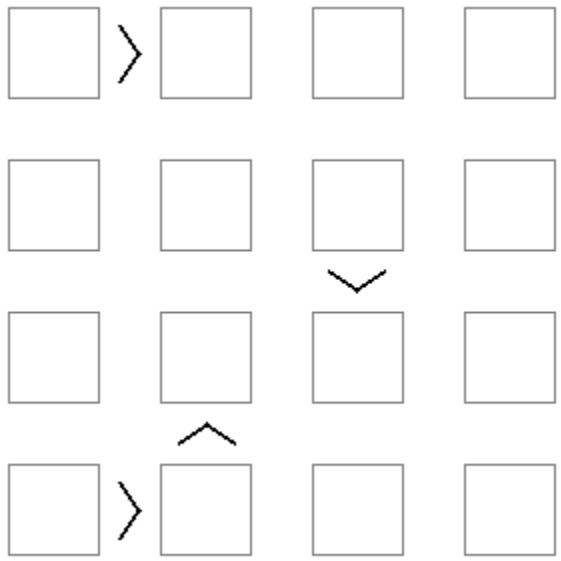
Udacity Take Home Assignment

Overview

You will write a solution validator for a board game similar to Sudoku. The game has an n by n board. Each row and column will be filled by numbers between 1 and n. Like Sudoku, each number can appear once and only once in each row and each column. But unlike Sudoku some adjacent cells have additional constraints between them. For example, in the board shown below cell (1, 1) should be greater than cell (1, 2), cell (2,3) should be greater than cell (3,3), and so on.

Example:



Here's an example of a valid solution with the constraints shown above:

Requirements

You will be writing a program to validate a given solution of the puzzle. You need to come up with the data structures to represent the board, the constraints and the solution, as well as the algorithm to validate it. For example here's the pseudo code:

```
boolean validate_solution(board, solution)
      return true if the solution is valid, false otherwise
```

For the purpose of creating a running program, you can hard code a 4x4 board with constraints, and valid/invalid solutions to validate. The constraints should be modeled as initial values in your data structure, as opposed to logic in your code (i.e. you cannot do things like `if cell(1,1) > cell(1,2) then …`)

Your solution must be executable and fully tested.

Develop your solution the same way as you do in your work and pay attention to all details.

Coding Conventions

You may use any language you choose and follow any coding convention as you wish.

Submission

Email us back your solution in a compressed file. The compressed file should include your code and all essential artifacts to execute your code such as data needed. You should also include instructions to build, test and run your program, and if applicable, workspaces for your IDE.

Alternatively, you may upload your solution to a publicly accessible github repo.

Grading Guidelines

Your submission will be graded with the following factors considered, among others:

- Correctness - whether the solution give correct results
- Completeness - whether the solution covers all scenarios, including handling of error conditions
- Design - Well structured, easily readable and maintainable
- Testability - Easy to test. Following principles of test driven development
- Style - Elegance of the the code and solution
- Timeliness of submission

Duration

We would like you to spend no more than 4 hours, cumulatively, on this exercise.