CS 245 - Assignment 1 Fall, 2017

Assignment 1 - Sudoku

The goal of this assignment is to demonstrate your mastery of recursion by generating a sudoku puzzle using a recursive algorithm.

Background

According to Wikipedia, Sudoku is a "logic-based combinatorial number-placement" game. In the classic and perhaps most popular version of the game, each Sudoku game board is a 9x9 grid with each cell in the grid filled with a single-number digit, 1 - 9, inclusive. In addition, each column, each row and each 3x3 subgrid must contain exactly one single-number digit with no repeats. A *subgrid* is sometimes called a *region* or *subregion*, a *box* or a *block*. An example of a Sudoku game board is shown below, with the subgrids divided by horizontal or vertical lines.

```
  -----------------------
| 9 1 2 | 7 8 3 | 5 4 6 |
| 3 4 5 | 9 1 6 | 7 8 2 |
| 6 7 8 | 2 4 5 | 3 9 1 |
  -----------------------
| 8 3 7 | 1 6 4 | 9 2 5 |
| 4 9 6 | 5 2 7 | 8 1 3 |
| 5 2 1 | 3 9 8 | 4 6 7 |
  -----------------------
| 7 6 4 | 8 3 1 | 2 5 9 |
| 1 5 9 | 4 7 2 | 6 3 8 |
| 2 8 3 | 6 5 9 | 1 7 4 |
  -----------------------
```

Figure 1: An example Sudoku game board

Although out of scope of this assignment, the game board you generate may be used later as a puzzle by suppressing the display of the contents of individual cells. The game player would then be required to determine the original contents of the cells.

Requirements

Your implementation will randomly generate a correct Sudoku game board using recursion and display the game board on the console. You may create a single class, Sudoku, for this solution.

Requirement 1: Your implementation must start by randomly generating one or more proposed numbers in the correct range (1-9) for each cell. If one of the randomly-generated proposed numbers has no conflicts in the row, in the column or in the subgrid, it may be placed on the Sudoku game board and the subsequent cells filled recursively. If your implementation is unable to find an appropriate proposal, your implementation must backtrack to find an appropriate number.

Requirement 2: Your implementation must print the correctly-generated Sudoku game board as shown in Figure 1.

Recommendations

Recommendation 1: Use a single class, Sudoku, for this implementation. This class may have multiple functions. Your implementation's "main" function may look like the following.

```
Sudoku sudoku = new Sudoku();

if (sudoku.fillBoard()) {
    sudoku.printBoard();
}
```

Optionally, since the "fillBoard()" function is expected to succeed, your implementation may look like the following instead:

```
Sudoku sudoku = new Sudoku();

sudoku.fillBoard();
sudoku.printBoard();
```

Recommendation 2: Your implementation may use a single 2-dimensional array for storing your implementation's solution. Likewise, your implementation may use a single variable — `size`, perhaps — to keep track of the size of the board.

Submission

Submit the source code for your implementation on Canvas. You may also add any comments in a README file (text, PDF or Word document) to help the grader understand or execute your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 75% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the

Implementation section above. Any deviation from the expected results results in 0 credit for implementation.

- 15% = Decomposition: in the eyes of the grader, your solution follow the suggestions above or otherwise must represent a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.

Late assignments will be accepted with a 10% penalty per day