Introduction to Computer Science II
Lab 7
Sudoku and Stacks

1 A Sudoku solver

Get the code: Start a *Terminal*. Then, grab some starting source code and open it:

```
$ cd
$ curl -L https://bit.ly/cosc-112-24f-lab-7 -o lab-7.zip
$ unzip lab-7.zip
$ cd lab-7
$ code .
```

You will find the following files:

- `Sudoku.java`: The class that contains the `main()` method. When run, it reads the puzzle from a file (specified by the user at the command line), displays the puzzle, calls the solver, and then displays the solution. Note that empty positions in the board are represented by a zero (`0`).
- `BoardPosition.java`: These objects are tied to a specific position on the puzzle (a.k.a., the board). When one is constructed, it finds and immutably sets itself to work with the first empty location (i.e., the first position, in scanner order[1], that contains a `0`). It can allow the value at its position to be set, it can determine whether the value at its position collides with another on the board,[2] print itself, etc.
- `easy.board`, `medium.board`, and `evil.board`: Three proper sudoku puzzles, given in the form that the `Sudoku` class can read. These files can be opened in a text editor and easily viewed or edited. Their difficulty—that is, the amount of backtracking required to solve them—is described by the name of each.

As provided, this code compiles and correctly solves these puzzles (and, I think, any other—feel free to add your own!).

[1] Left-to-right, top-to-bottom.

[2] That is, does the value at its position also occur in its row, column, or subgrid.

Your task: Modify the `solve()` method in `Sudoku.java` **not** to use recursion. Instead, your solver must create and maintain its own *stack* to direct the search for a solution (which should be driven by a loop). You may, of course, add classes and methods as needed. I also recommend using the standard Java `Stack<E>` class.[3]

2 How to submit your work

Submit your `NiceLinkedList.java` file by uploading it into the `lab-7` folder in your shared Google Drive folder for this course.

This assignment is due on Sunday, Nov-17, 11:59 pm.

[3] Or if you are adventurous, create your own `NiceStack<E>` interface, and then implement a `NiceArrayStack<E>` or `NiceLinkedStack<E>` class. Then write your solution to use that!