EECE 2560: Fundamentals of Engineering Algorithms
Department of Electrical and Computer Engineering

Project #4

Write a program that solves Sudoku puzzles. The input to Sudoku is a 9x9 board that is subdivided into 3x3 squares. Each cell is either blank or contains an integer from 1 to 9.

A solution to a puzzle is the same board with every blank cell filled in with a digit from 1 to 9 such that every digit appears exactly once in every row, column, and square.

The input to the program is a text file containing a collection of Sudoku boards, with one board per line. For example:

```
.....2.......7...17..3...9.8..7......2.89.6...13..6....9..5.824.....891.........
3...8.......7....51.............36...2..4....7...........6.13..452...........8..Z
```

For each board that is read, the output is a printout of the board correctly filled in.

4. reset cell (i, j)
{ get the value at cell (i, j)
} update conflicts.

Part a

Some of the declarations and definitions for the `board` class are given to you. (Optional. You can choose to write your own codes.) Add functions to the class that:

3: set cell (i, j, value)
{ set cell (i, j) w/ value.
} update conflicts

(1)
1. initialize the board, and update conflicts, ✓
2. print the board and the conflicts to the screen, ✓
3. add a value to a cell, and update conflicts, set
4. clear a cell, and update conflicts, and reset
5. check to see if the board has been solved (return true or false, and print the result to the screen)

IsSolved ( ) ✓

For each row i and digit j, keep track of whether each digit j has been placed in row i. Do the same for each column and each square. We will use this information in part b of the project to write the Sudoku solver.

The code you submit should read each Sudoku board from the file one-by-one, print the board and conflicts to the screen, and check to see if the board has been solved (all boards will not be solved at this point).

two data members: { board_item. n×n
conflicts. n×n×n