Jose Garay
HW8
CS325

Puzzle: Sudoku
Description: A number puzzle where the player must fill in all spaces in the 9x9 grid. A number placed in a square may be a number between 1 and 9, a number cannot be repeated in the same column or row, and a number cannot be repeated within it's subgrid.

Summarization: For verifying user input, I can verify a nxn sudoku puzzle in O(n) time for any puzzle in my program. I randomly generate a puzzle at runtime and a prerequisite for generating a puzzle is generating a complete sudoku puzzle. By storing the complete solution, I can just compare the user's input to the solution to verify user input.

To verify a sudoku puzzle without having the solution beforehand would take O(n) and consist of 3 for loops:

1.) Check each element in every row making sure that no number is duplicated and all numbers are between 1 and n.
2.) Check each element in every column making sure that no number is duplicated and all numbers are between 1 and n.
3.) Check each element in every sub-grid making sure that no number is duplicated and all numbers are between 1 and n.

NP-Proof
The question for the sudoku puzzle was "Given a Sudoku instance, does it have any solutions?". We must prove that this problem is in NP and also that this problem is NP-Hard to conclude that this problem is in NP-Complete.
NP
For a given filled out sudoku grid, the certificate, we can verify that it is a valid sudoku puzzle solution in polynomial time. To verify a sudoku grid, we must ensure that it complies to the three rules of sudoku:

- All rows must contain numbers 1-9 with no duplicates
- All columns must contain numbers 1-9 with no duplicates.
- All subgrids must contain numbers 1-9 with no duplicates.

Each one of these operations is O(n) and since these operations happen separate from each other, the complexity for all of them is also O(n).

NP-Complete
Given that 3-SAT is NP-Complete, we can reduce 3-SAT to sudoku. 3-SAT has 3 clauses ($\mathbf{c_1, c_2, c_3}$) which can be related to sudoku's three constraints of are all rows, columns, and subgrids valid/unique($\mathbf{c_1}`, \mathbf{c_2}`, \mathbf{c_3}`$)? Within each clause the variables **($\mathbf{x_1, x_2, x_3}$…)** can be related to sudoku **($\mathbf{x_1}`, \mathbf{x_2}`, \ \mathbf{x_3}`$, ...)** checking if a specific row, column, or subgrid is valid/unique. If 3-SAT is satisfiable, then sudoku is satisfiable. Therefore, 3-SAT reduces to sudoku and sudoku is NP-Complete.