- Sudoku -

The goal of Sudoku is to fill a 9×9 grid with numbers so that each row, column and 3×3 section contain all of the digits between 1 and 9. At the beginning of the game, the 9×9 grid will have some of the squares filled in. The player should fill in the missing digits and complete the grid. A move is incorrect if:

- Any row contains more than one of the same numbers from 1 to 9.
- Any column contains more than one of the same numbers from 1 to 9.
- Any 3×3 grid contains more than one of the same numbers from 1 to 9.

In this project, you are asked to implement Sudoku using concepts of object oriented and backtracking. The project should contain following items:

- Three levels of difficulties:
  - Easy (30 unfilled cells)
  - Medium (40 unfilled cells)
  - Difficult (50 unfilled cells).
- Generating random Sudoku grids. Created Sudokus (for any level of difficulty) should be solvable. So don't generate the Sudoku by mere randomness. Use backtracking to generate solvable Sudoku.
- Solving given Sudoku. The program should be able to solve Sudokus given by user or those generated randomly. Use backtracking for solving the Sudoku. If there is no solution, announce it to the user.
- A GUI for showing the Sudoku grid to the player and let the player fill the grid. It should check whether the player's solution is a valid solution or not. Announce the player as winner or loser.
- An option for loading unsolved Sudoku maps which are stored in files. You can use filedialog from tkinter.

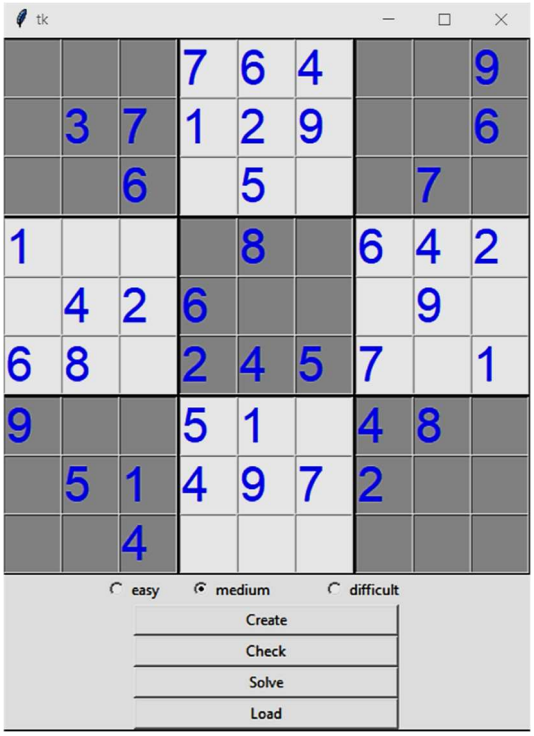
Implementing backtracking for generating/solving Sudoku has 5 points. Implementing the GUI has 2 points. The whole project has 7 (out of 20) points. Implementing without OOP has only 4 points.

Be aware that any student who cheats in the project will fail the course with 0.

Screenshots for a sample solution is given in the appendix.

Appendix:

- The GUI for the Sudoku program:



- Sample solution: