Assignment 3
CSP to solve Sudoko

1 Game Description

Sudoku is a logic-based number-placement game that challenges players to fill a 9x9 grid with digits from 1 to 9. The objective is to complete the grid in such a way that each row, each column, and each of the nine 3x3 subgrids (also known as regions or boxes) contains all of the digits from 1 to 9 without repetition.

7	9			1	3	6		
4				7		3		
1			2	4		9	7	5
5			6			2		7
	7				1	8		
8		6	9	2		5		
6		1			2		5	3
3						4		9
	2	4		3	5			

2 Requirements

2.1 Game GUI

- Mode 1: You are required to provide a full game with GUI to show the AI agent solving the game.
- Mode 2 : game with GUI which allow user to input board representation then agent solves it

2.2 Algorithms

You are required to support Backtracking to be able to validate the input(to check that input puzzle is solvable) , Backtracking is also needed to generate random puzzle (fill random places of puzzle) to ensure that the puzzle generated is solvable.

2.3 Arc Consistency

You are required to SHOW Arc consistency for Solution

- Represent Sudoku as a CSP: Variables: Each cell in the Sudoku grid is a variable. Domains: The domain of each variable represents the possible numbers (1 to 9) that can be placed in the cell. Constraints: The Sudoku rules, which state that no number can be repeated in a row, column, or 3x3 subgrid.
- Define Arcs: An arc in Sudoku represents a binary constraint between two variables (cells). Arc consistency is applied to all pairs of connected variables.

  For each row, create arcs between all pairs of cells in that row. For each column, create arcs between all pairs of cells in that column. For each 3x3 subgrid, create arcs between all pairs of cells in that subgrid.
- Initial Domain Reduction: Before applying arc consistency, initialize the domains of each variable based on the initial puzzle.

  For each pre-filled cell, remove all other values from its domain. For each empty cell, initialize its domain to [1, 2, 3, 4, 5, 6, 7, 8, 9].
- Apply Arc Consistency: Iteratively enforce arc consistency on all arcs until no further changes can be made:

  For each arc (Xi, Xj): Revise: For each value in the domain of Xi, check if there is a consistent value in the domain of Xj. If inconsistent: Remove the inconsistent value from the domain of Xi. Repeat for all arcs: Continue revising all arcs until no further changes can be made.
- Update Sudoku Grid: After applying arc consistency, update the Sudoku grid based on the reduced domains:

  For each cell with a singleton domain (a domain with only one value), assign that value to the cell.
- Repeat: Continue the process of enforcing arc consistency and updating the Sudoku grid until no more changes can be made and all board is filled .

3 Notes

3.1 Deliverables

- Your well commented code.
- A report showing your work,
  including:
  - sample runs and their corresponding Arc consistency trees
  - comparison between different initial boards (easy, intermediate, hard) and the time needed to solve puzzle.

Also the report should contain the data structures used (if any) and algorithms, Assumptions and details you find them necessary to be clarified, Any extra work.

3.2 Bonus

- There will be a bonus if game was interactive(make user fill board and check if input correct or violate constraints for each number) and working correctly.

3.3 Further Notes

- You may use Java, Python or C++ for your implementation.
- Copied assignments will be severely penalized.
- You can work in groups of 2 or 3.
- You will be evaluated individually in discussion

Good Luck