**Assignment 2**

**Implement Constraint Satisfaction Problem for Sudoku Solver**

**Theory -**

A Constraint Satisfaction Problem (CSP) is a powerful problem-solving framework used to find a solution that satisfies a set of given conditions or constraints. CSPs are formally defined by a set of variables, a domain of possible values for each variable, and a set of constraints on these variables.

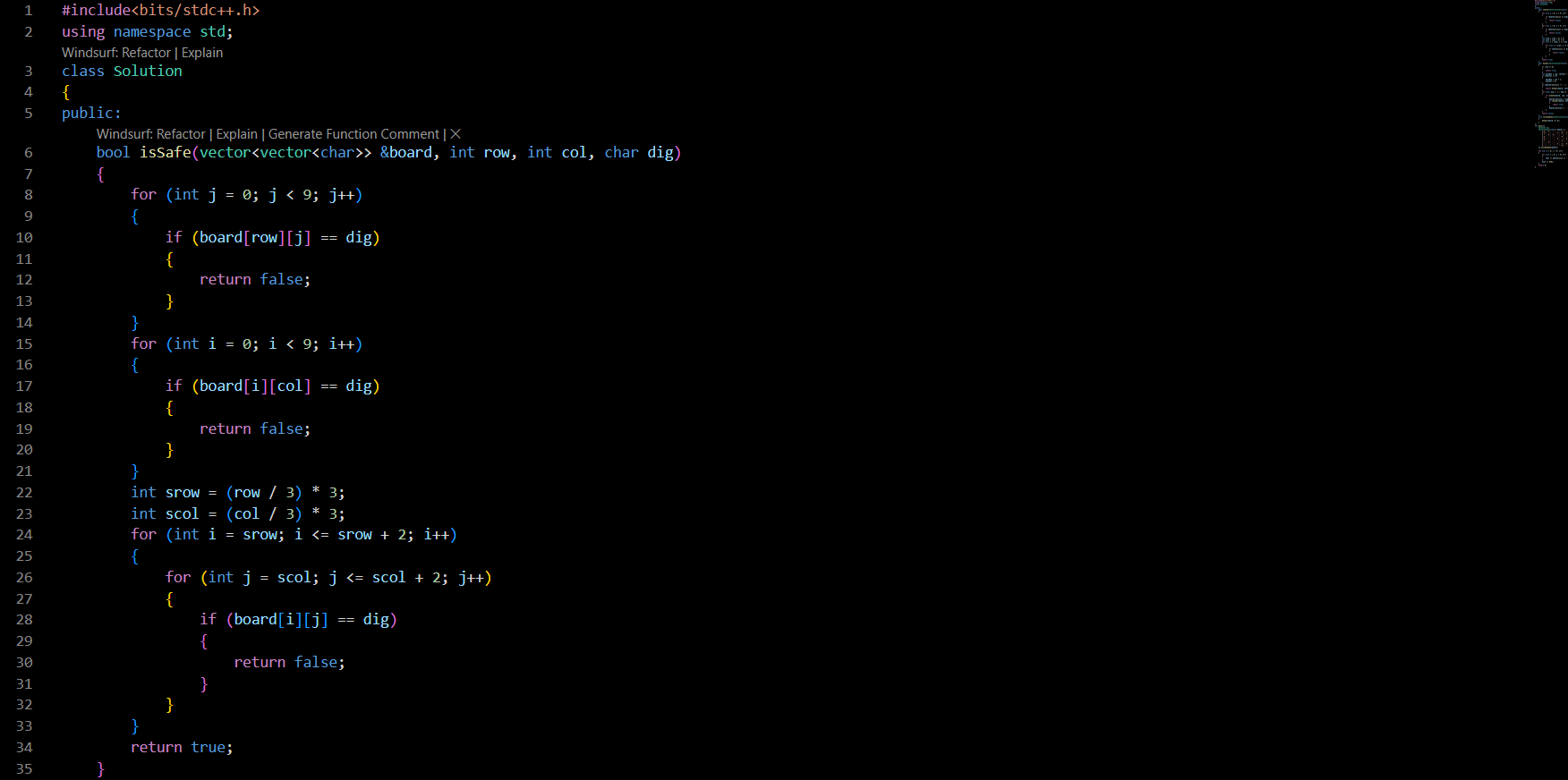
The most common algorithm for solving CSPs is backtracking search. This algorithm incrementally builds a solution by assigning values to variables one by one. If an assignment violates a constraint, the algorithm backtracks to the previous variable and tries a different value.

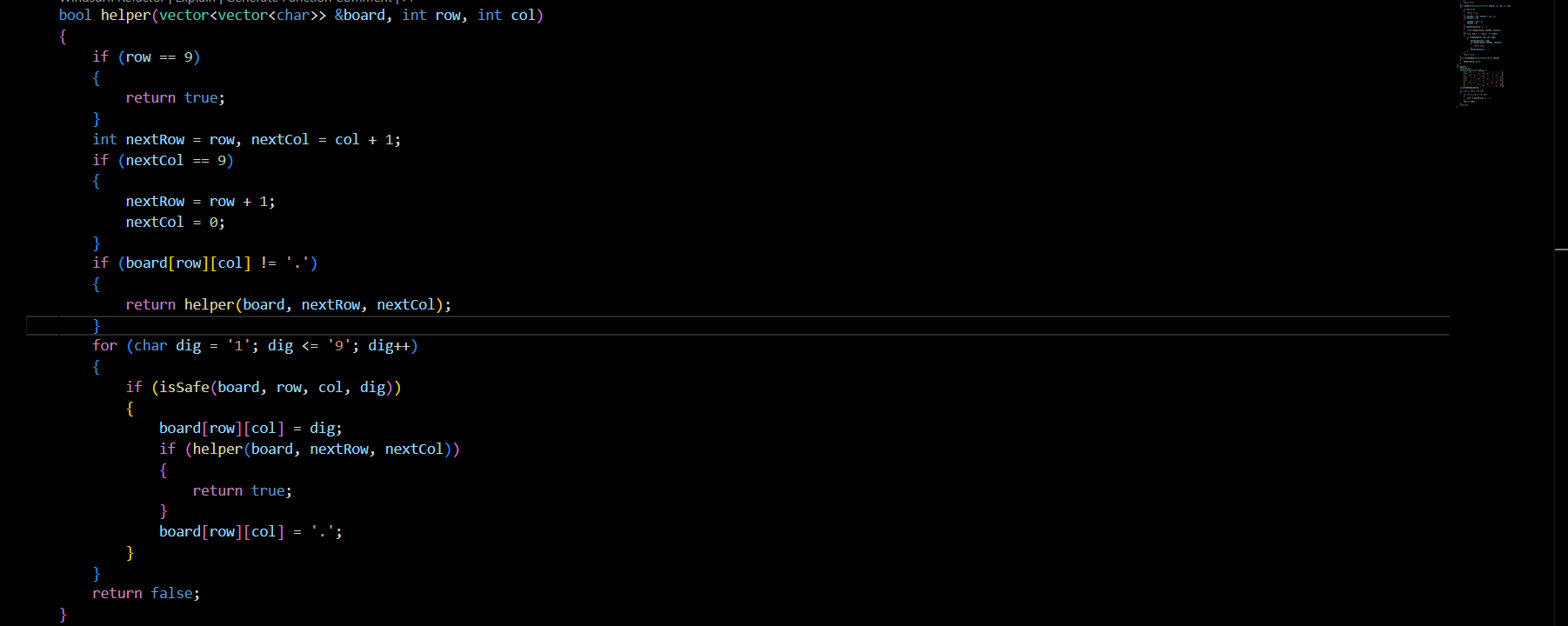
Applying CSP to Sudoku: Sudoku can be naturally modelled as a CSP:

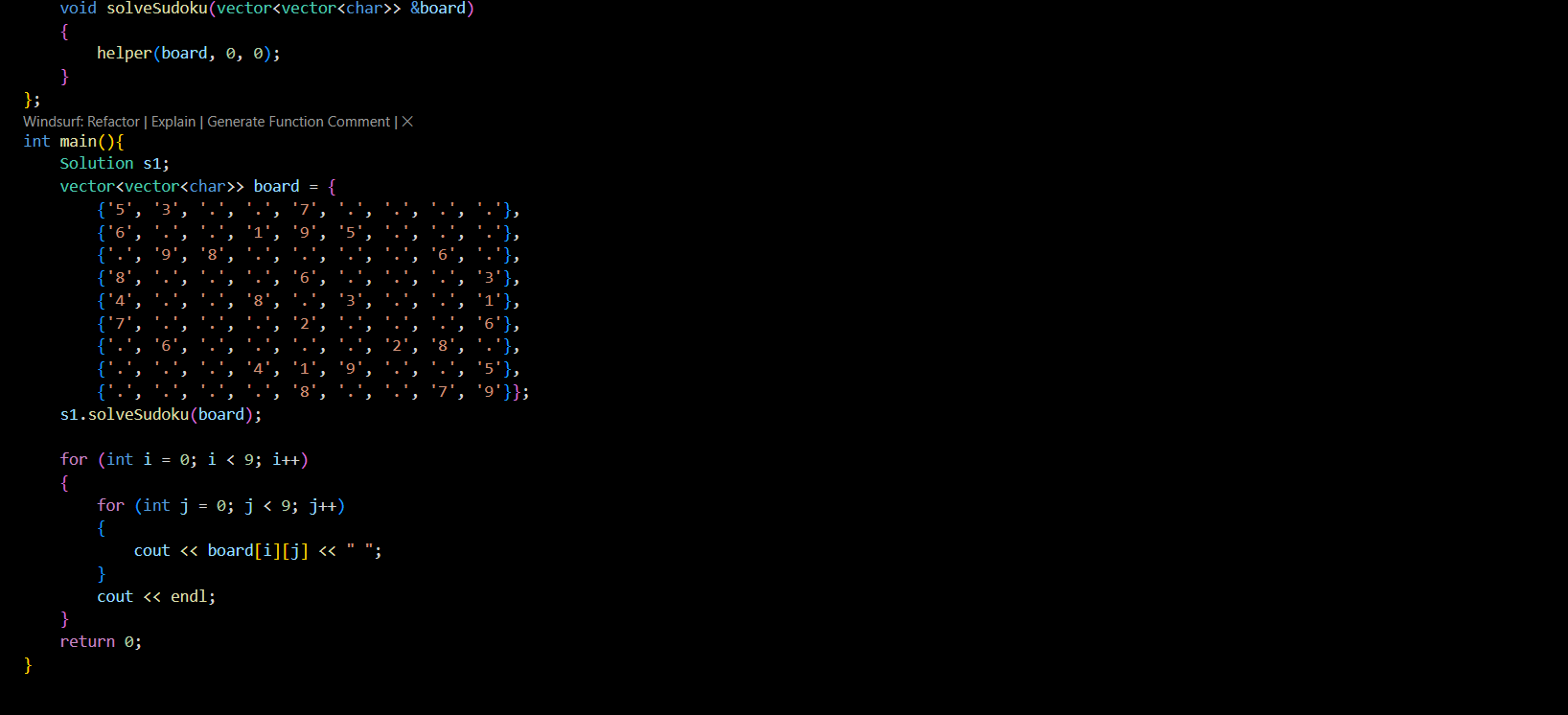
* Variables (X): The 81 cells on the Sudoku grid, which can be represented as for row and column .
* Domains (D): For each empty cell, the domain is the set of possible numbers .
* Constraints (C): The rules of Sudoku define the constraints:
  1. Row Constraint: All cells in the same row must have unique values.
  2. Column Constraint: All cells in the same column must have unique values.
  3. 3x3 Box Constraint: All cells within t he same 3x3 sub grid must have unique values.

The goal is to find a complete assignment of values to all cell variables from their domains such that all three constraints are met.

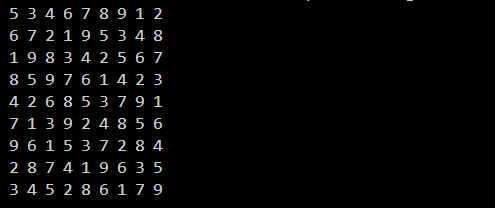
**Code –**

****

****

****

**Output –**

****

**Explanation –**

1. Representation of the Puzzle: The Sudoku board is represented as a 9x9 2D array or matrix, where 0 denotes an empty (unassigned) cell. This grid directly maps to the variables in our CSP model.
2. Backtracking Algorithm: The solveSudoku function implements the backtracking algorithm. It works by finding the first empty cell and trying to fill it with a valid number (from 1 to 9).
3. Constraint Checking: The isSafe() helper function is crucial for enforcing the constraints. Before placing a number, it checks if that number already exists in the current row, column, or 3x3 subgrid. This ensures that no rules of Sudoku are violated with the new assignment.
4. Recursive Search and Backtracking: If a number is deemed "safe," it is placed on the board, and the solveSudoku function calls itself recursively to solve the rest of the puzzle. If the subsequent recursive calls fail to find a solution (i.e., return false), it means the current number placement was incorrect. The algorithm then backtracks by resetting the cell to 0 and tries the next number.
5. Solving and Output: This process continues until a complete and valid assignment for all 81 cells is found, at which point the base case is met, and the solution is returned. If the algorithm tries all possibilities and cannot find a solution, it will report that the puzzle is unsolvable.

**Conclusion -**

We have successfully modeled the Sudoku puzzle as a Constraint Satisfaction Problem and outlined a backtracking algorithm to find its solution by systematically satisfying all row, column, and box constraints.