**Topic 6.3: Sudoku Puzzle Solver**

**Question**  
Write a program to solve a Sudoku puzzle by filling the empty cells.  
A Sudoku solution must satisfy all of the following rules:

* Each of the digits 1–9 must occur exactly once in each row.
* Each of the digits 1–9 must occur exactly once in each column.
* Each of the digits 1–9 must occur exactly once in each of the 9 sub-boxes of the grid.  
  The '.' character indicates empty cells.

**Aim**  
To implement a backtracking algorithm to solve a partially filled Sudoku board, ensuring all Sudoku rules are satisfied.

**Algorithm**

1. Start with the given Sudoku board.
2. Find the first empty cell.
3. Attempt to place digits 1–9 in that cell.
4. For each placement, check if it is valid according to Sudoku rules:
   * The number is not already in the same row.
   * The number is not already in the same column.
   * The number is not already in the corresponding 3×3 sub-box.
5. If valid, place the number and recursively attempt to solve the next cell.
6. If no number fits, backtrack and reset the cell to empty, then try another possibility.
7. Continue until all cells are filled.

**Output**A screenshot of a computer

AI-generated content may be incorrect.

**Result**  
The Sudoku puzzle was successfully solved by applying backtracking and constraint checking.

**Performance Analysis**

* Time Complexity: O(9^(n)) in the worst case, where n is the number of empty cells.
* Space Complexity: O(1) (constant board size 9×9) plus recursion stack depth O(n).