**Name:** Satyam Jaiswal **UID:** 2021600028 **Batch**: B **Date:** 15/11/2023

## EXP 10: Constraint Satisfaction Problem

**Problem:** Sudoku solver

1- define variables

2-Domain

3- constraints

4-constraint propagation

5- resulting reduced domain of the variables

6- solving suduko with backtracking and considering reduced domains

**Program:**

def print\_board(board):

for row in board:

print(" ".join(map(str, row)))

def is\_valid(board, row, col, num):

# Check if the number is not in the same row or column

for i in range(9):

if board[row][i] == num or board[i][col] == num:

return False

# Check if the number is not in the same 3x3 grid

start\_row, start\_col = 3 \* (row // 3), 3 \* (col // 3)

for i in range(3):

for j in range(3):

if board[start\_row + i][start\_col + j] == num:

return False

return True

def solve\_sudoku(board):

for row in range(9):

for col in range(9):

if board[row][col] == 0:

for num in range(1, 10):

if is\_valid(board, row, col, num):

board[row][col] = num

print\_board(board)

print("\n")

if solve\_sudoku(board):

return True

board[row][col] = 0

return False # Backtrack if no number is valid

return True # Puzzle is solved

# Example Sudoku board (0 represents empty cells)

sudoku\_board = [

[5, 3, 0, 0, 7, 0, 0, 0, 0],

[6, 0, 0, 1, 9, 5, 0, 0, 0],

[0, 9, 8, 0, 0, 0, 0, 6, 0],

[8, 0, 0, 0, 6, 0, 0, 0, 3],

[4, 0, 0, 8, 0, 3, 0, 0, 1],

[7, 0, 0, 0, 2, 0, 0, 0, 6],

[0, 6, 0, 0, 0, 0, 2, 8, 0],

[0, 0, 0, 4, 1, 9, 0, 0, 5],

[0, 0, 0, 0, 8, 0, 0, 7, 9]

]

print("Initial Sudoku Board:")

print\_board(sudoku\_board)

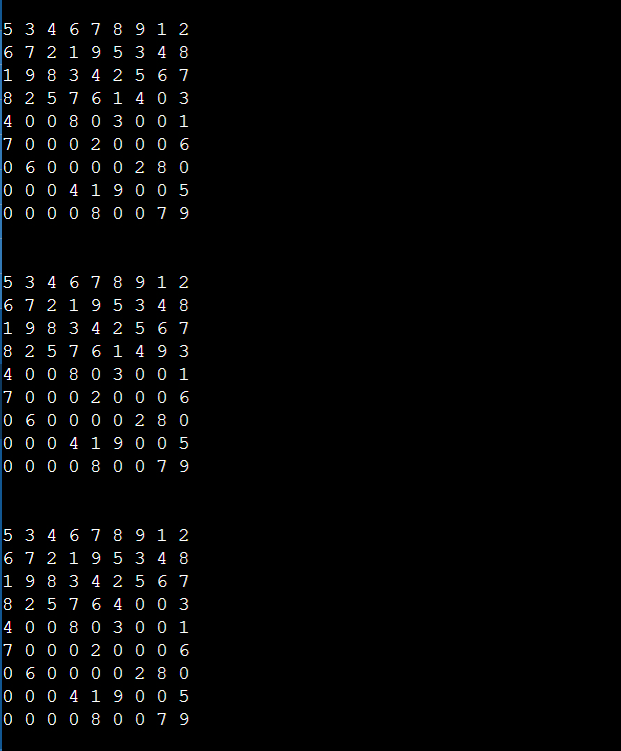
print("\nSolving Sudoku...\n")

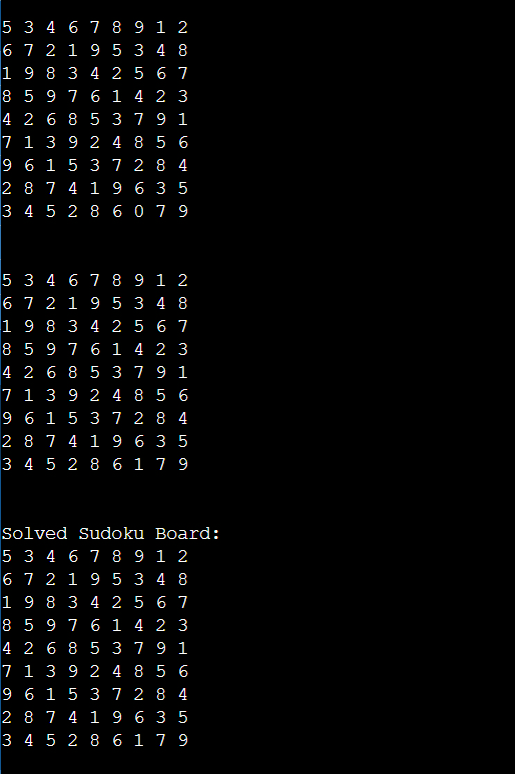
solve\_sudoku(sudoku\_board)

print("Solved Sudoku Board:")

print\_board(sudoku\_board)

**Output**:

****

****