**Experiment 1**

**a ) Write a program to solve 8 Queens problem**

**Aim:**

To implement the 8-Queens problem using backtracking and find all possible arrangements of 8 queens on an 8×8 chessboard such that no two queens attack each other.

**Procedure:**

1. Initialize an 8×8 chessboard with zeros.
2. Define a function to check if placing a queen is safe.
3. Use backtracking to place queens column by column.
4. Backtrack if a valid placement is not possible.
5. Print all valid solutions or indicate if none exist.

Top of Form

**Program:**

N = 8

def printSolution(board):

for row in board:

print(" ".join("Q" if cell else "." for cell in row))

print("\n")

def isSafe(board, row, col):

for i in range(col):

if board[row][i]:

return False

for i, j in zip(range(row, -1, -1), range(col, -1, -1)):

if board[i][j]:

return False

for i, j in zip(range(row, N, 1), range(col, -1, -1)):

if board[i][j]:

return False

return True

def solveNQueens(board, col):

if col >= N:

printSolution(board)

return True

res = False

for i in range(N):

if isSafe(board, i, col):

board[i][col] = 1

res = solveNQueens(board, col + 1) or res

board[i][col] = 0

return res

def solve():

board = [[0] \* N for \_ in range(N)]

if not solveNQueens(board, 0):

print("No solution found")

solve()

**Output:**

[[0, 0, 0, 0, 0, 0, 0, 1],

[0, 0, 0, 0, 1, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 1, 0],

[0, 0, 1, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 1, 0, 0],

[0, 1, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 1, 0, 0, 0, 0],

[1, 0, 0, 0, 0, 0, 0, 0]]

**Result:**

The program successfully finds and prints all valid solutions for the 8-Queens problem