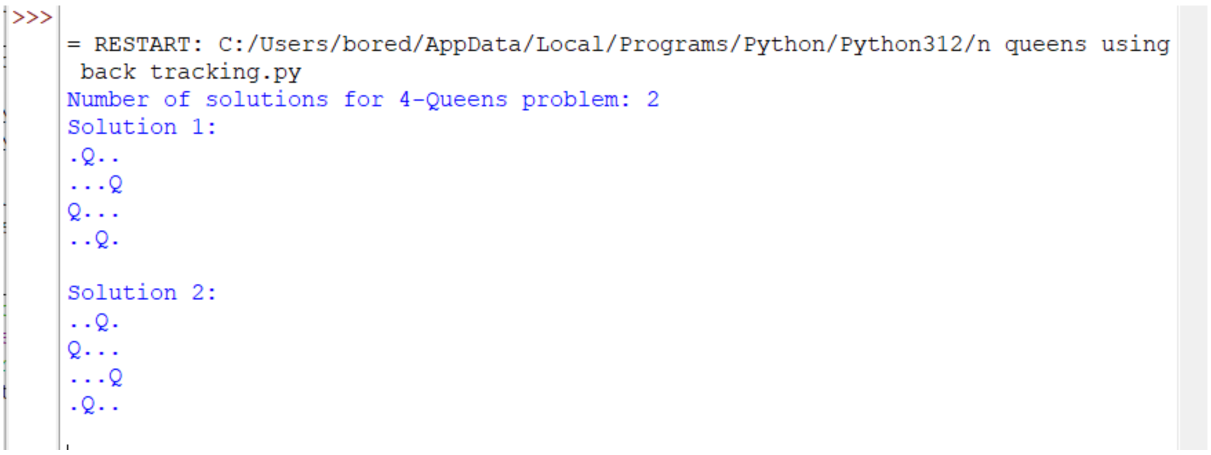
LAB-14

N queens using back tracking

CODE:

```
def is_safe(board, row, col, N):
    for i in range(row):
        if board[i][col] == 1:
            return False
    i, j = row, col
    while i >= 0 and j >= 0:
        if board[i][j] == 1:
            return False
        i -= 1
        j -= 1
    i, j = row, col
    while i >= 0 and j < N:
        if board[i][j] == 1:
            return False
        i -= 1
        j += 1
    return True


def solve_n_queens_util(board, row, N, solutions):


    if row == N:


        solutions.append(["".join("Q" if col == 1 else "." for col in row) for row in board])
        return


    for col in range(N):
        if is_safe(board, row, col, N):
            board[row][col] = 1
            solve_n_queens_util(board, row + 1, N, solutions)
            board[row][col] = 0

def solve_n_queens(N):
    board = [[0] * N for _ in range(N)]
    solutions = []
    solve_n_queens_util(board, 0, N, solutions)
    return solutions
N = 6
solutions = solve_n_queens(N)
print(f"Number of solutions for {N}-Queens problem: {len(solutions)}")
for i, solution in enumerate(solutions, 1):
    print(f"Solution {i}:")
    for row in solution:
        print(row)
    print()
```

OUTPUT:



```
>>>
= RESTART: C:/Users/bored/AppData/Local/Programs/Python/Python312/n queens using
 back tracking.py
Number of solutions for 4-Queens problem: 2
Solution 1:
.Q..
...Q
Q...
..Q.

Solution 2:
..Q.
Q...
...Q
.Q..
```