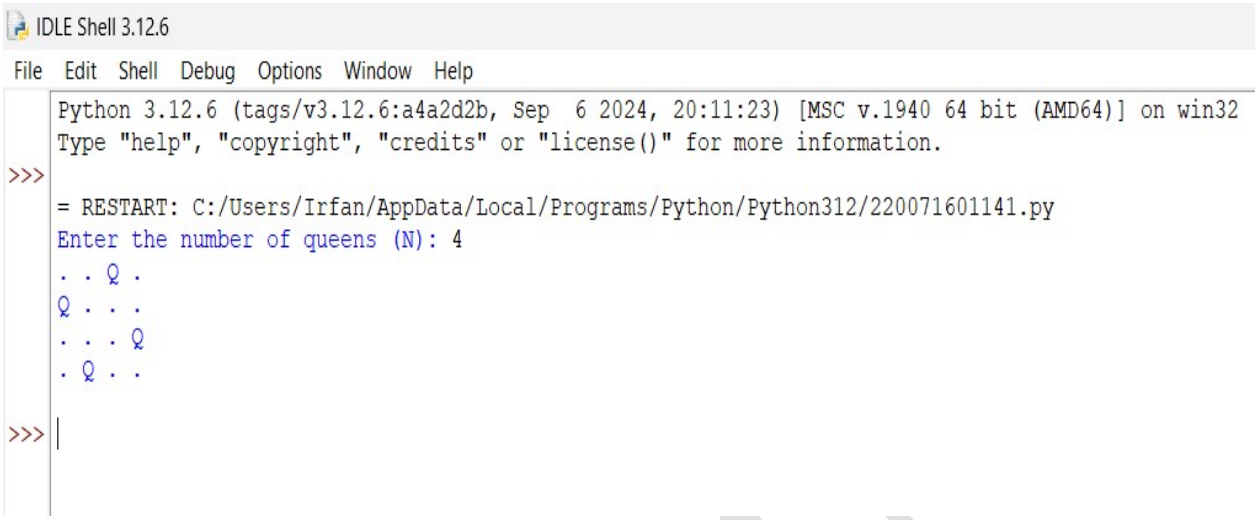
N QUEENS PROBLEM

AIM:

ALGORITHM:

OUTPUT:



PROGRAM:

```
def print_board(board):
    for row in board:
        print(" ".join("Q" if cell else "." for cell in row))
    print()
def is_safe(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, len(board)), range(col, -1, -1)):
        if board[i][j]:
            return False
    return True
def solve_n_queens(board, col):
    if col >= len(board):
        return True
    for i in range(len(board)):
     if is_safe(board, i, col):
            board[i][col] = True
            if solve_n_queens(board, col + 1):
                return True
            board[i][col] = Falsereturn False
```

```
def n_queens(n):
    board = [[False] * n for _ in range(n)]
    if solve_n_queens(board, 0):
        print_board(board)
    else:
        print("No solution exists")
if __name__== "_main_":
    try:
        n = int(input("Enter the number of queens (N): "))
        if n <= 0:
            print("Number of queens must be a positive integer.")
        else:
            n_queens(n)
    except ValueError:
        print("Invalid input. Please enter a positive integer.")
```

RESULT: