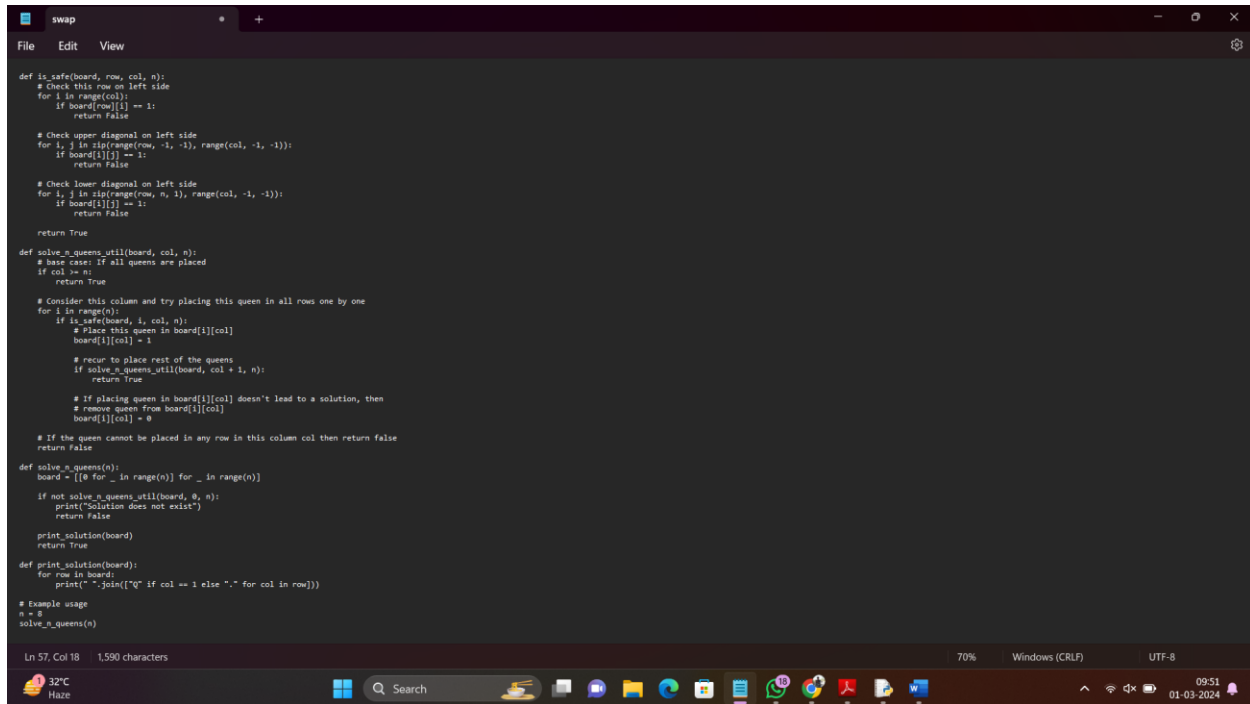


AIM:

Write the python program to solve 8-Queen problem

PROGRAM:

A screenshot of a Python IDE window titled 'swap'. The code defines a function 'is_safe' to check if a queen can be placed at a given position without conflicting with others already on the board. It then defines 'solve_n_queens_util' as a recursive helper function that tries to place a queen in each row of a column and recurses on the next column. The main 'solve_n_queens' function initializes the board and calls the utility function. Finally, a 'print_solution' function is defined to format and print the board state. At the bottom, an example usage is shown with n=8.

```
def is_safe(board, row, col, n):
    # Check this row on left side
    for i in range(col):
        if board[row][i] == 1:
            return False

    # Check upper diagonal on left side
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check lower diagonal on left side
    for i, j in zip(range(row, n, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solve_n_queens_util(board, col, n):
    # base case: If all queens are placed
    if col == n:
        return True

    # Consider this column and try placing this queen in all rows one by one
    for i in range(n):
        if is_safe(board, i, col, n):
            # place this queen in board[i][col]
            board[i][col] = 1

            # recur to place rest of the queens
            if solve_n_queens_util(board, col + 1, n):
                return True

            # If placing queen in board[i][col] doesn't lead to a solution, then
            # remove queen from board[i][col]
            board[i][col] = 0

    # If the queen cannot be placed in any row in this column col then return false
    return False

def solve_n_queens(n):
    board = [[0 for _ in range(n)] for _ in range(n)]

    if not solve_n_queens_util(board, 0, n):
        print("Solution does not exist")
        return False

    print_solution(board)
    return True

def print_solution(board):
    for row in board:
        print(" ".join(["Q" if col == 1 else "." for col in row]))

# Example usage
n = 8
solve_n_queens(n)
```

INPUT: Enter the size of the board (e.g., 8 for the 8-Queens problem): 8

OUTPUT: Total solutions found: 92

Q.....

....Q..

....Q..

.....Q

.Q.....

...Q....

.....Q.

..Q.....

s(continues to print all 92 solutions for the 8-Queens problem)

RESULT: 8-Queen problem successfully executed and verified.