

PROGRAM-2

8 QUEENS PROBLEM

AIM:-

To write and execute the python program for the 8 queens program.

PROCEDURE:-

- **Check Safety:**
 - The `is_safe` function checks whether placing a queen at a specific position on the board is safe. It checks for conflicts with queens placed in previous columns and diagonals.
- **Backtracking Algorithm:**
 - The `solve_n_queens_util` function implements a backtracking algorithm to find a solution to the N-Queens problem. It recursively explores all possible configurations of queens on the board, ensuring that each placement is safe.
- **Main Function:**
 - The `solve_n_queens` function initializes the chessboard and calls the `solve_n_queens_util` function to find a solution. If a solution exists, it prints the board; otherwise, it prints "Solution does not exist."
- **Print Board Function:**
 - The `print_board` function prints the chessboard configuration with queens represented as '1's and empty squares represented as '0's.
- **Example Usage:**
 - Call the `solve_n_queens` function with the desired board size (N) to find a solution for the N-Queens problem.

CODING:-

```
def is_safe(board, row, col, N):
```

```
    for i in range(col):
```

```
        if board[row][i] == 1:
```

```
            return False
```

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

```
    if board[i][j] == 1:
```

```
        return False
```

```
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):
```

```
    if col >= N:
```

```
        return True
```

```
    for i in range(N):
```

```
        if is_safe(board, i, col, N):
```

```
            board[i][col] = 1
```

```
            if solve_n_queens_util(board, col + 1, N):
```

```
                return True
```

```
            board[i][col] = 0
```

```
    return False
```

```
def solve_n_queens(N):
```

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

```
if not solve_n_queens_util(board, 0, N):
```

```
    print("Solution does not exist")
```

```
    return False
```

```
print("Solution exists and is as follows:")
```

```
print_board(board)
```

```
return True
```

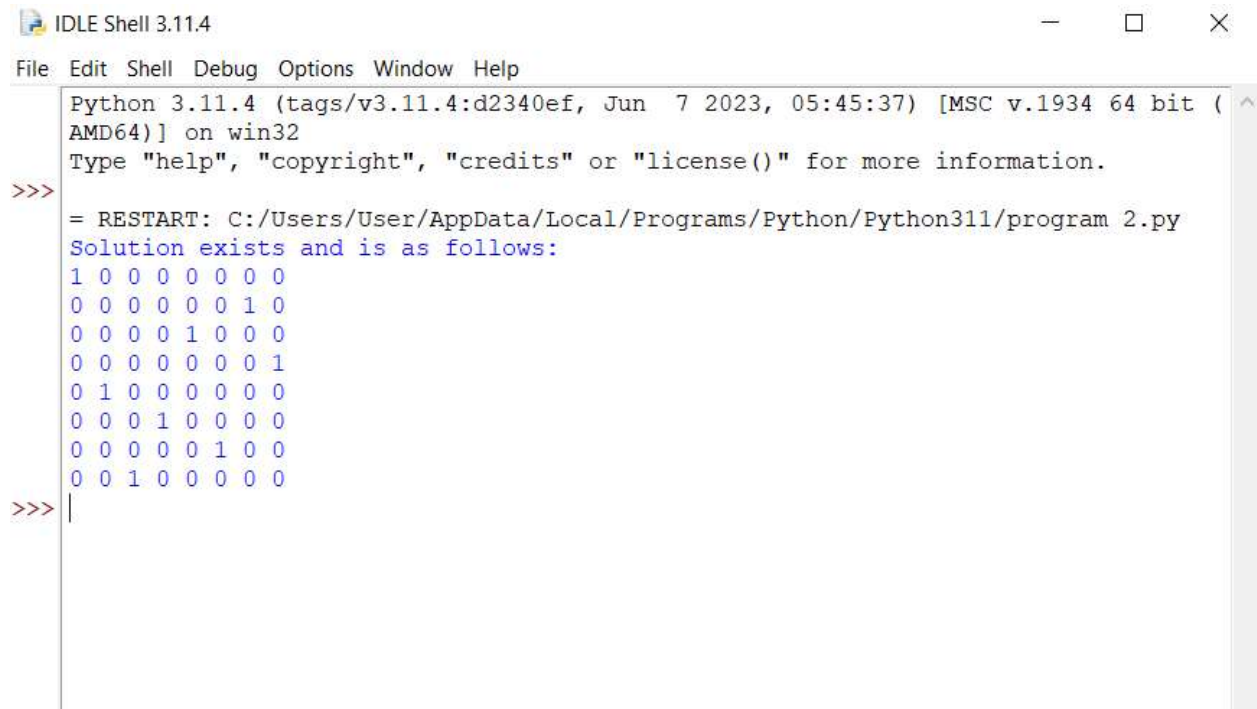
```
def print_board(board):
```

```
    for row in board:
```

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

```
solve_n_queens(8)
```

OUTPUT:-

A screenshot of the IDLE Shell 3.11.4 window. The title bar reads 'IDLE Shell 3.11.4'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The shell area shows the following text: 'Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', and a prompt '>>>'. Below the prompt, the text '= RESTART: C:/Users/User/AppData/Local/Programs/Python/Python311/program 2.py' is displayed. This is followed by the message 'Solution exists and is as follows:' and a 10x10 grid of numbers. The grid contains 1s at positions (0,0), (0,9), (1,9), (2,8), (3,7), (4,6), (5,5), (6,4), (7,3), (8,2), and (9,1), with all other positions being 0. The prompt '>>>' is shown again at the bottom left of the shell area.

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/User/AppData/Local/Programs/Python/Python311/program 2.py
Solution exists and is as follows:
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
>>>
```

RESULT:-

Hence the program has been successfully executed and verified.