

**EX.NO:3**

**DATE:4/9/2024**

**Reg.no:220701013**

## **8-QUEENS PROBLEM**

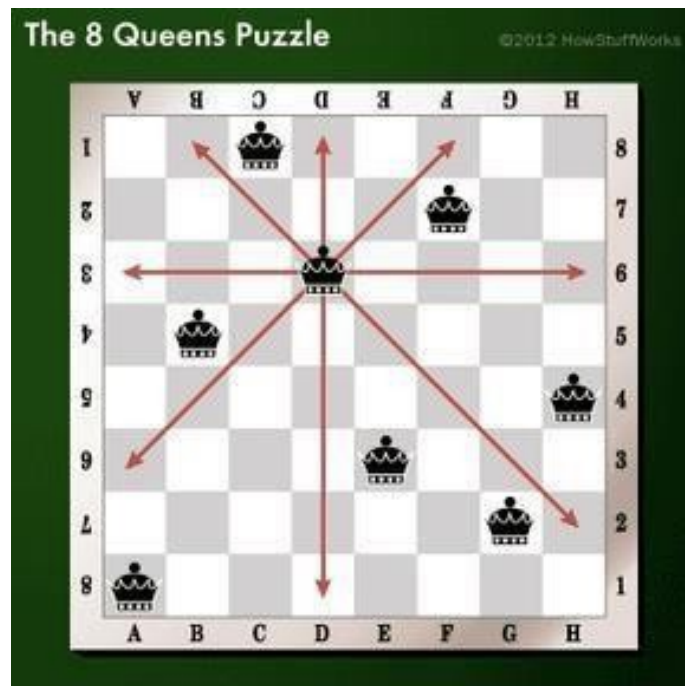
### **AIM:**

To implement an 8-Queens problem using Python.

You are given an 8x8 board; find a way to place 8 queens such that no queen can attack any other queen on the chessboard. A queen can only be attacked if it lies on the same row, same column, or the same diagonal as any other queen.

Print all the possible configurations.

To solve this problem, we will make use of the Backtracking algorithm. The backtracking algorithm, in general checks all possible configurations and test whether the required result is obtained or not. For the given problem, we will explore all possible positions the queens can be relatively placed at. The solution will be correct when the number of placed queens = 8.



## CODE:

```
def printSolution(board):
    for i in range(len(board)):
        for j in range(len(board)):
            print(board[i][j], end=' ')
        print()

def isSafe(board, row, col):
    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, len(board), 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solveNQutil(board, col):
    if col >= len(board):
        return True

    for i in range(len(board)):
        if isSafe(board, i, col):

            board[i][col] = 1

            if solveNQutil(board, col + 1):
                return True

            board[i][col] = 0
    return False

def solveNQ():
    def solveNQ():
        N = int(input("Enter the size of the board (N): "))

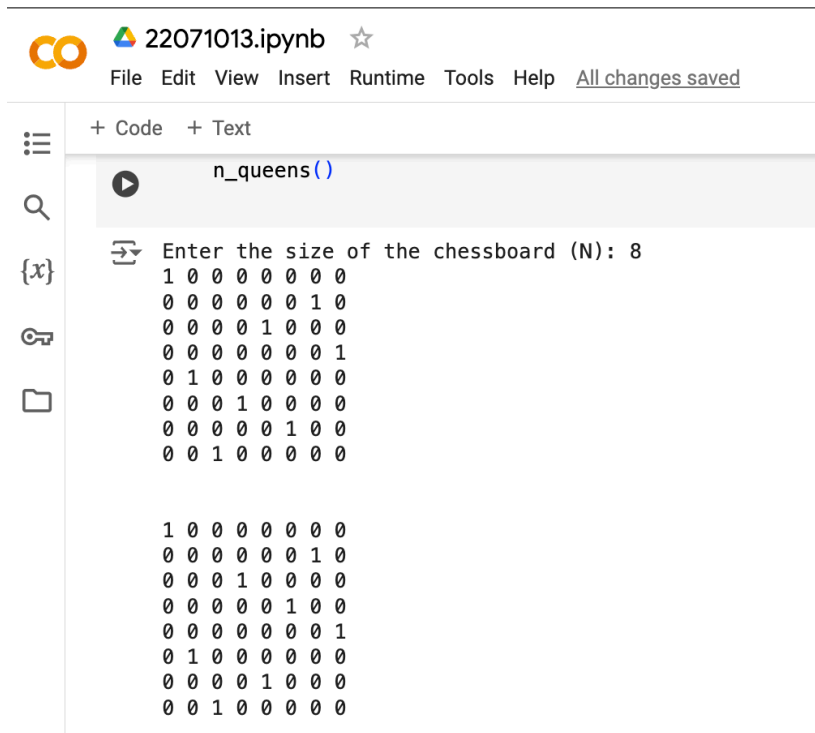
        board = [[0 for _ in range(N)] for _ in range(N)]

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

        printSolution(board)
        return True

    solveNQ()
```

## OUTPUT:



The image shows a Jupyter Notebook interface with the following components:

- Header:** A logo with the letters 'CO' in orange, followed by the text '22071013.ipynb' and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a link 'All changes saved'.
- Left Sidebar:** Contains icons for a list, search, a variable '{x}', a key, and a folder.
- Main Area:**
  - At the top, there are tabs for '+ Code' and '+ Text', and a cell titled 'n\_queens()' with a play button icon.
  - Below the cell, the output is displayed. It starts with the prompt 'Enter the size of the chessboard (N): 8'.
  - The output then shows two 8x8 matrices of 0s and 1s, representing solutions to the 8-Queens problem. Each matrix has exactly one '1' in each row and each column.

```
Enter the size of the chessboard (N): 8
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

1 0 0 0 0 0 0 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 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
```

## RESULT:

Thus, the 8-Queens program has been implemented successfully.