

EX.NO:1

DATE:11/9/2024

Reg.no:220701010

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

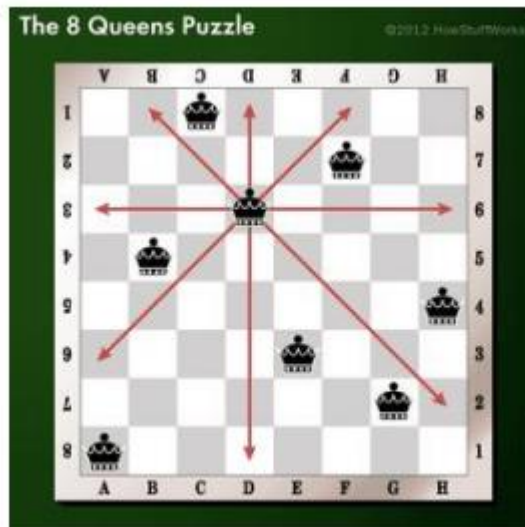
queen on the chessboard. A queen can only be attacked if it lies on the same row, same column,

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 print_board(board):
    for row in board:
        print(' '.join('Q' if cell else '.' for cell in row))
    print()

def is_safe(board, row, col, N):
    # Check this column
    for i in range(row):
        if board[i][col]:
            return False

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

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

    return True

def solve_nqueens_util(board, row, N):
    if row >= N:
        print_board(board)
        return True

    res = False
    for col in range(N):
        if is_safe(board, row, col, N):
            board[row][col] = True
            res = solve_nqueens_util(board, row + 1, N) or res
            board[row][col] = False

    return res

def solve_nqueens(N):
    board = [[False] * N for _ in range(N)]
    if not solve_nqueens_util(board, 0, N):
        print("Solution does not exist")
N=int(input("enter n: "))
solve_nqueens(N)
```

OUTPUT:

The screenshot displays a Jupyter Notebook environment. At the top, the title bar reads "220701010-NQUEEN.ipynb" with a star icon. Below it is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help. A toolbar contains icons for code, text, and execution. The notebook content is divided into two sections: a code cell and a text cell. The code cell contains the following Python code:

```
print("solution does not exist")
N=Int(input("enter n: "))
solve_nqueens(N)
```

The text cell shows the output of the code, which is a series of 8x8 grids representing solutions for N=8. Each grid has 'Q' at the position of a queen. The first few solutions shown are:

```
enter n: 8
Q . . . . .
. . . Q . .
. . . . . Q
. . . . . Q .
. Q . . . .
. . . . . Q
. Q . . . .
. . Q . . .

Q . . . . .
. . . . . Q
. . Q . . .
. . . . . Q
. . Q . . .
. Q . . . .
. . . . . Q
. . . . . Q

Q . . . . .
. . . . . Q
. . . . . Q
. . . . . Q
. Q . . . .
. . . . . Q
. . . . . Q
. . . . . Q

Q . . . . .
. . . . . Q
. . . . . Q
. . . . . Q
. Q . . . .
. . . . . Q
. . . . . Q
. . . . . Q
```

The interface also includes a sidebar on the left with icons for file explorer, search, and other tools. On the right, there is a "Connect" button and a "Gemin" button. At the bottom right, there is a watermark that says "Activate Windows Go to Settings to activate Windows."