Exp No:01
Date:

N - Queens

Aim:

To write an algorithm and code for an N-Queens problem such that no two queens attack each other in N×N Chess board.

Algorithm:

Step-1 : Start

Step 2: Initialize the board. Create an N×N chessboard

Step 3: Define the function to check if it the queens are placed successfully and print the board

Step 4: Define the function to check if the queens are placed Successfully and print the board

Step 5: In the function, check if the queen is placed safely and for each column in current row.

Step 6: Initialize the N×N board and call the recursive function

Step 7: Stop

Program:

```
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), range (col, -1, -1)):
    if board [i][j] == 1:
        return false.
return True
def solve_n queens (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 (board, col+1, n):
                return True
            board [i][col] = 0
            return false
def Print_board (board, n)
    for i in range (n):
        for j in range (n)
            Print ("Q" if board [i][j] == 1 el
                                "." , end = " ")
            Print ()
def n_queens (n):
    board = [[0 for _ in range (n)] for _ in
                                range (n)]
    if not solve_n queens (board, 0, n):
```

def ts

```
        Print ("Solution does not exist")
            return
        Print_board ( board, n)
    n = int ( input ("Enter the Size of the board:")
    n - queens ()
```

Output

Enter the Size of the board 4

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

Result:

Thus the program for N-queens has been successfully executed