EXP NO: 1

DATE :

N-QUEENS PROBLEM

AIM:

To implement N-queens problem using python.

ALGORITHM:

1) Initialize the board as an NxN grid filled with 0 (empty).

2) Define a function (is_safe) to check if placing a queen in a row and column is safe:

→ check if there's a queen in the same row or diagonals on the left side.

3) Define the main function (solve_n_queens_util):

→ If all queens are placed, return true.

→ For each row in the current column, check if it's safe to place a queen:

* If safe, place the queen and try to place the next one recursively.

* If this placement doesn't work, backtrack by removing the queen.

4) Solve the N-queens problem using the utility function. If no solution exists, print "Solution does not exist"; otherwise, print the board.

5) Print the solution if found.

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
    return True
def solve_n_queens(N):
    board = [[0]*N for _ in range(N)]
    if solve_n_queens_util(board, 0, N) == False:
        print("Solutions does not exist")
        return False
    print_solution(board)
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) == True:
```

```
        return True
    board[i][col] = 0
return False
def print_solution(board):
    N = len(board)
    for i in range(N):
        for j in range(N):
            print(board[i][j], end = " ")
        print()
N = int(input("Enter a number "))
solve_n-queens(N)
```

OUTPUT:

Enter a number 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

RESULT:

Thus the python code has been implemented to solve 8 queens problem.