**Experiment No: 10**

**Name:Shubham Kaspate**

**Roll No:20141241**

**Batch:I3**

# Implement 8-queens problem using backtracking - Code -

#include <bits/stdc++.h>

using namespace std;

int board[8][8];

bool isPossible(int n, int row, int col)

{

    for (int i = row - 1; i >= 0; i--)

    {

        if (board[i][col] == 1)

        {

            return false;

        }

    }

    for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--)

    {

        if (board[i][j] == 1)

        {

            return false;

        }

    }

    for (int i = row - 1, j = col + 1; i >= 0 && j < n; i--, j++)

    {

        if (board[i][j] == 1)

        {

            return false;

        }

    }

    return true;

}

void nQueenHelper(int n, int row)

{

    if (row == n)

    {

        for (int i = 0; i < n; i++)

        {

            for (int j = 0; j < n; j++)

            {

                cout << board[i][j] << " ";

            }

        }

        cout << endl;

        return;

    }

    for (int j = 0; j < n; j++)

    {

        if (isPossible(n, row, j))

        {                             // if no attack, proceed

            board[row][j] = 1;        // mark row, column with 1

            nQueenHelper(n, row + 1); // call function to continue

        }

        board[row][j] = 0;

    }

    return;

}

void placeNQueens(int n)

{

    memset(board, 0, 8 \* 8 \* sizeof(int)); // allocate 8\*8 memory

    nQueenHelper(n, 0);                    // call the backtracking function

}

int main()

{

    int n;

    cin >> n;

    placeNQueens(n);

    return 0;

}

# Output –

