# Name : Shrinivas Hatyalikar

# Div: TY-B (B2)

# Roll No: 26

**Q) Implement N Queens Problem using backtracking.**

**Code:**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Solution {

public:

    bool isSafe(int row, int col, vector<string>& board, int n) {

        // Check Upper Diagonal

        int currrow = row;

        int currcol = col;

        while (row >= 0 && col >= 0) {

            if (board[row][col] == 'Q')

                return false;

            row--;

            col--;

        }

        row = currrow;

        col = currcol;

        // Check for rows

        while (col >= 0) {

            if (board[row][col] == 'Q')

                return false;

            col--;

        }

        row = currrow;

        col = currcol;

        // Check Lower Diagonal

        while (row < n && col >= 0) {

            if (board[row][col] == 'Q')

                return false;

            row++;

            col--;

        }

        return true;

    }

    void solve(int col, vector<string>& board, vector<vector<string>>& ans, int n) {

        if (col == n) {

            ans.push\_back(board);

            return;

        }

        for (int row = 0; row < n; row++) {

            if (isSafe(row, col, board, n)) {

                board[row][col] = 'Q';

                solve(col + 1, board, ans, n);

                board[row][col] = '.';

            }

        }

    }

    vector<vector<string>> solveNQueens(int n) {

        vector<string> board(n, string(n, '.'));

        vector<vector<string>> ans;

        solve(0, board, ans, n);

        return ans;

    }

};

void printChessboard(const vector<string>& board) {

    for (const string& row : board) {

        cout <<  row  << endl;

    }

    cout << endl;

}

int main() {

    cout << "Enter Size of Grid: ";

    int n;

    cin >> n;

    Solution solution;

    vector<vector<string>> result = solution.solveNQueens(n);

    // Printing the result

    for (const vector<string>& board : result) {

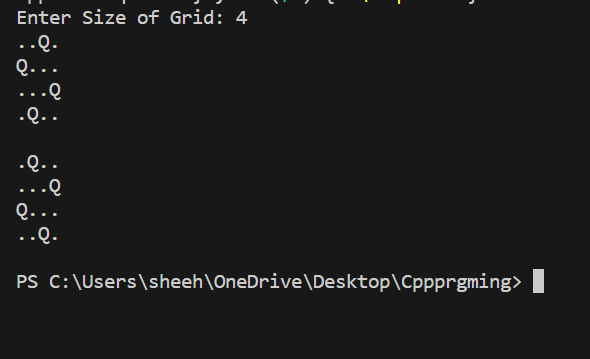
        printChessboard(board);

    }

    return 0;

}

**Output:**

****