VISHWAKARMA INSTITUTE OF TECHNOLOGY

COMPUTER ENGINEERING

Name: Abhinav Mahajan

Division: TY-C

Roll No: 15

Subject: Artificial Intelligence (AI)

<u>LAB ASSIGNMENT NO – 4</u>

Implementation of **N-Queens Problem** as a Constraint Satisfaction Problem.

The N-Queens problem is a classic Constraint Satisfaction Problem (CSP) where you need to place N queens on an N×N chessboard in such a way that no two queens threaten each other. This means no two queens can be in the same row, column, or diagonal.

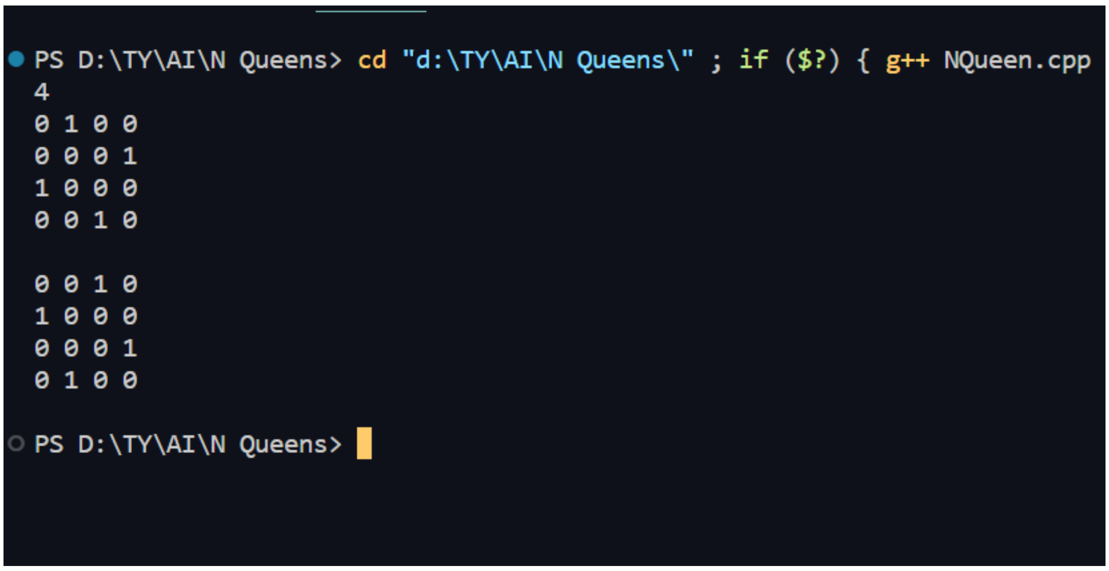
Approach - Backtracking:

1. Begin with the first column, which represents a variable.
2. For each row in the current column's domain:
   a. Check if placing a queen in this row violates any of the established constraints. If it does, proceed to the next row.
   b. When a valid row is found, assign it to the current column and move on to the next column.
   c. If you reach a column where it's impossible to place a queen without breaking the constraints, backtrack to the previous column and explore the next row.
3. Keep following this process until you've successfully placed N queens on the chessboard, achieving a solution, or determine that no solution exists.

Code:

```cpp
#include<bits/stdc++.h>
using namespace std;

bool isSafe(int col, int row, int n, vector<vector<int>>& v){
    for(int i = row-1; i >= 0; i--){
        if(v[i][col] == 1) return false;
    }

    for(int i = row-1, j = col-1; i >= 0 && j >= 0; i--, j--){
        if(v[i][j] == 1) return false;
    }

    for(int i = row-1, j = col+1; i >= 0 && j < n; i--, j++){
        if(v[i][j] == 1) return false;
    }
    return true;
}

void placeNQueens(int n, int row, vector<vector<int>>& v){
    if(row == n){
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                cout << v[i][j] << " ";
            }
            cout << endl;
        }
        cout << endl;
        return;
    }

    for(int j = 0; j < n; j++){
        if(isSafe(j, row, n, v)){
            v[row][j] = 1;
            placeNQueens(n, row+1, v);
            v[row][j] = 0;
        }
    }
}

int main(){

    int n;
    cin >> n;
    vector<vector<int>> v(n, vector<int>(n, 0));
    if(n < 4){
        cout << "Solution does not exist for this input!" << endl;
        return 0;
    }

    placeNQueens(n, 0, v);
    return 0;
}
```

Output:



```
● PS D:\TY\AI\N Queens> cd "d:\TY\AI\N Queens\" ; if ($?) { g++ NQueen.cpp
  4
  0 1 0 0
  0 0 0 1
  1 0 0 0
  0 0 1 0

  0 0 1 0
  1 0 0 0
  0 0 0 1
  0 1 0 0

○ PS D:\TY\AI\N Queens> 
```