Name: Shaan Agarwal
UID: 2021300001
Subject: Design And Analysis Of Algorithms (DAA)
Title: Experiment 07

Aim: Implement N Queen Problem Using Backtracking

Theory:

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.

Algorithm

1. Start by placing a queen in the first column of the first row.
2. Move to the next column and try to place a queen in the first row. If this is possible, move to the next column and repeat. If it is not possible, move to the next row in the same column and try again.
3. Continue this process until all N queens have been placed on the board or until it is determined that no valid configuration is possible.

4. If a valid configuration has been found, output the locations of the queens on the board. If no valid configuration is found, output an appropriate message.

This basic algorithm can be implemented recursively by checking all possible configurations of queens for each column. The key to this approach is the use of backtracking, which involves undoing previous choices when they lead to a dead end and exploring alternative choices.

PseudoCode

```
procedure solveNQueens(board, col):
    if col = N then
        output board
        return true
    for row from 0 to N-1 do
        if isSafe(board, row, col) then
            board[row][col] = 1
            if solveNQueens(board, col+1) then
```

```
            return true
        board[row][col] = 0
    return false

function isSafe(board, row, col):
    for i from 0 to col-1 do
        if board[row][i] = 1 then
            return false
    for i from row, j from col down to 0 do
        if board[i][j] = 1 then
            return false
    for i from row, j from col down to 0 and i < N do
        if board[i][j] = 1 then
            return false
    return true

initialize empty N x N board
solveNQueens(board, 0)
```

Program

```cpp
#include <iostream>
#include <vector>

using namespace std;

bool isSafe(vector<vector<int>>& board, int row, int col)
{
    int N = board.size();
    for (int x = 0; x < col; x++)
        if (board[row][x] == 1)
            return false;
    for (int x = row, y = col; x >= 0 && y >= 0; x--, y--)
        if (board[x][y] == 1)
            return false;
    for (int x = row, y = col; x < N && y >= 0; x++, y--)
        if (board[x][y] == 1)
            return false;
    return true;
}

int solveNQueens(vector<vector<int>>& board, int col, int& count, int& steps)
{
    int N = board.size();
    if (col == N) {
        count++;
        if (count == 1 || count == 2) {
            cout << "Solution " << count << ":" << endl;
            for (int i = 0; i < N; i++) {
                for (int j = 0; j < N; j++)
                    cout << board[i][j] << " ";
                cout << endl;
            }
            cout << endl;
        }
        return count;
    }
    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 1;
            steps++;
            count = solveNQueens(board, col + 1, count, steps);
            board[i][col] = 0;
        }
    }
```

```
    return count;
}

int main()
{
    int N;
    cout << "Enter the value of N: ";
    cin >> N;
    vector<vector<int>> board(N, vector<int>(N, 0));
    int count = 0, steps = 0;
    int final_count = solveNQueens(board, 0, count, steps);
    if (final_count == 0)
        cout << "No solution found";
    else
        cout << "Number of solutions found: " << final_count << " in " <<
steps << " steps" << endl;
    return 0;
}
```

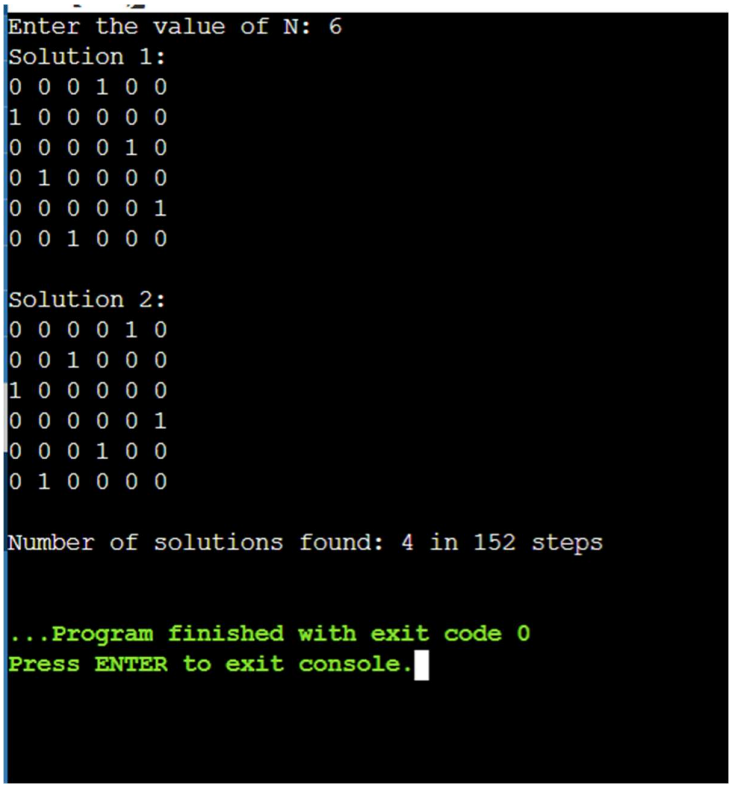
Output

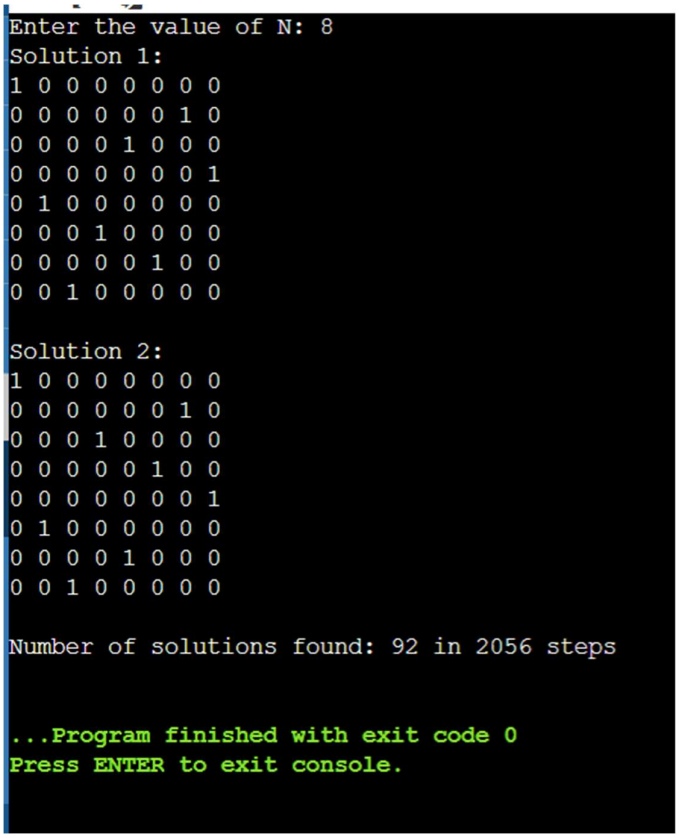
```
Enter the value of N: 4
Solution 1:
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0

Solution 2:
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0

Number of solutions found: 2 in 16 steps


...Program finished with exit code 0
Press ENTER to exit console.
```


Enter the value of N: 6
Solution 1:
0 0 0 1 0 0
1 0 0 0 0 0
0 0 0 0 1 0
0 1 0 0 0 0
0 0 0 0 0 1
0 0 1 0 0 0
Solution 2:
0 0 0 0 1 0
0 0 1 0 0 0
1 0 0 0 0 0
0 0 0 0 0 1
0 0 0 1 0 0
0 1 0 0 0 0
Number of solutions found: 4 in 152 steps
...Program finished with exit code 0
Press ENTER to exit console.

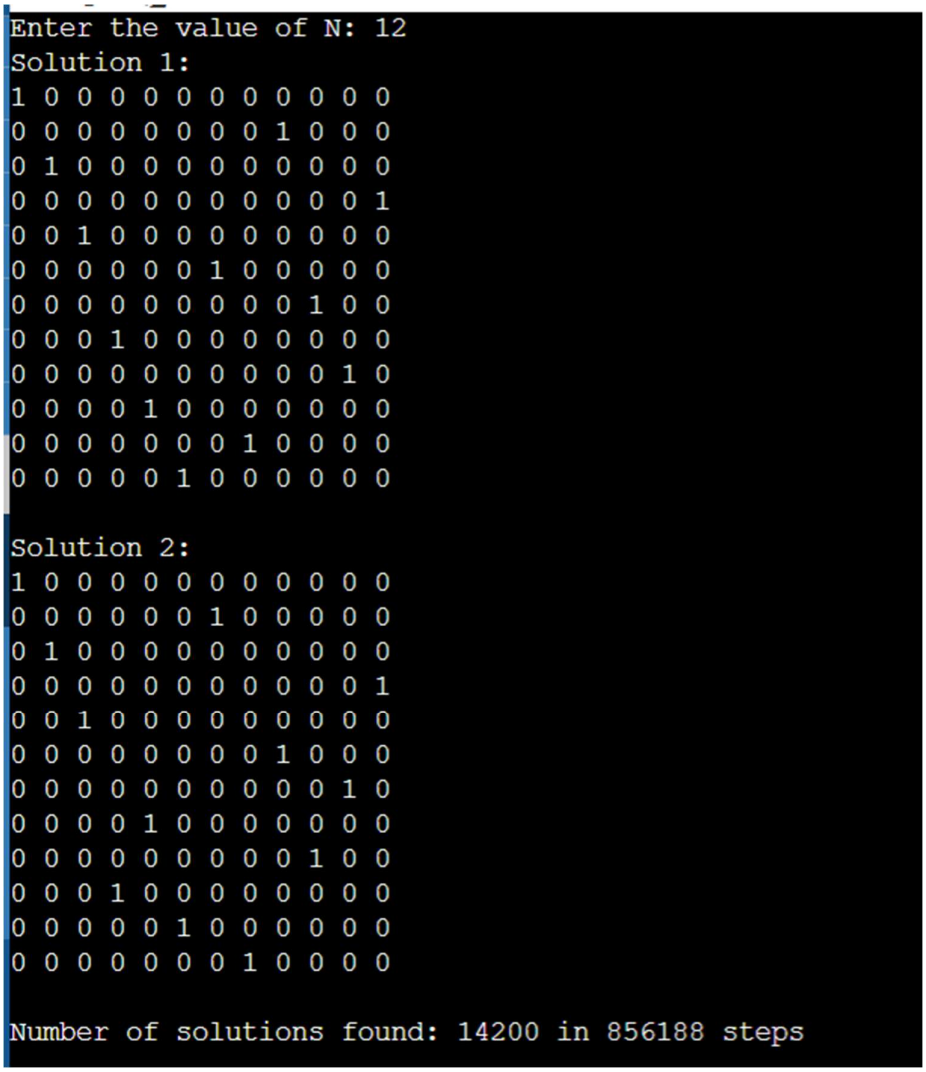

Enter the value of N: 8
Solution 1:
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
Solution 2:
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
Number of solutions found: 92 in 2056 steps
...Program finished with exit code 0
Press ENTER to exit console.

```
Enter the value of N: 10
Solution 1:
1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 0

Solution 2:
1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0

Number of solutions found: 724 in 35538 steps


...Program finished with exit code 0
Press ENTER to exit console.
```



Conclusion: After performing the above experiment, I have understood the concept of backtracking and have implemented the same for solving N Queen Problem using C++.