Experiment Number 7

Name ::	Rishabh Anand	UID ::	19BCS4525
Branch ::	CSE - IoT	Sec/Grp ::	1/A
Semester ::	5th	Date ::	14th Nov, 2021
Subject ::	Adv Programming Lab	CODE ::	CSP-347

1. Aim :

Implement N Queen's problem using Back Tracking.

2. Task :

1. Implement N Queen's problem using Back Tracking.

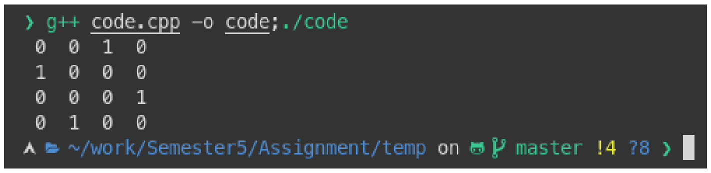
3. Algorithm :

- Start in the leftmost column
- If all queens are placed return true
- Try all rows in the current column. Do following for every tried row.
  - If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.
  - If placing the queen in [row, column] leads to a solution then return true.
  - If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.
- If all rows have been tried and nothing worked, return false to trigger backtracking.

4. Source Code :

```
#define N 4
#include <bits/stdc++.h>

void printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
            printf(" %d ", board[i][j]);
        printf("\n");
    }
}

bool isSafe(int board[N][N], int row, int col)
{
    int i, j;
    for (i = 0; i < col; i++)
        if (board[row][i])
            return false;
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
            return false;
    for (i = row, j = col; j >= 0 && i < N; i++, j--)
        if (board[i][j])
            return false;
    return true;
}

bool solveNQUtil(int board[N][N], int col)
{
    if (col >= N)
        return true;
    for (int i = 0; i < N; i++)
    {
        if (isSafe(board, i, col))
        {
            board[i][col] = 1;
            if (solveNQUtil(board, col + 1))
                return true;
            board[i][col] = 0;
        }
    }
    return false;
}

bool solveNQ()
{
    int board[N][N] = {{0, 0, 0, 0},
                       {0, 0, 0, 0},
                       {0, 0, 0, 0},
                       {0, 0, 0, 0}};
    if (solveNQUtil(board, 0) == false)
    {
        printf("Solution does not exist");
        return false;
    }
    printSolution(board);
    return true;
}

int main()
{
    solveNQ();
    return 0;
}
```

5. Observations :



```
❯ g++ code.cpp -o code;./code
 0  0  1  0
 1  0  0  0
 0  0  0  1
 0  1  0  0
~/work/Semester5/Assignment/temp on  master !4 ?8 ❯
```

Learning Outcomes :

- Come to know about graph and its concepts of indegree outdegree etc.
- Come to know about graphs and its different types and their classification on the basis of different parameters.
- Understand the N-queen problem and solve by using backtracking and also knew some parameters about the problem.

S. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			