Experiment - 7

Student Name: Pankaj Singh Kanyal
UID: 20BCS6668
Branch: AIML
Section/Group: AIML 4 B
Semester: 5th
Date of Performance: / /2022
Subject Name: Advanced Programming Lab
Subject Code: 20CSP-334

1. AIM:

Implement N Queens problem using Backtracking.

2. Apparatus:

- Texeditor
- Laptop / PC with C++ compiler

3. Algorithm/Theory

Backtracking Algorithm

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes, then we backtrack and return false.

1) Start in the leftmost column

2) If all queens are placed return true

3) Try all rows in the current column.

Do the following for every tried row.

a) 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.

b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

4) If all rows have been tried and nothing worked, return false to trigger backtracking.

4. Program/Code

```
#include <bits/stdc++.h>
#define N 5
using namespace std;

void printSolution(int board[N][N])
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << " " << 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, 0, 0, 0 , 0},
                        { 0, 0, 0, 0, 0} };

    if (solveNQUtil(board, 0) == false) {
        cout << "Solution does not exist";
        return false;
    }

    printSolution(board);
    return true;
}
int main()
{
    solveNQ();
    return 0;
}
```

6. Output

```
→  Advance Programming Lab cd "/media/pankajsingh/workbench/Semester 5/Advance Programming Lab/Experiment_7/" && g++ nquees.
nquees && "/media/pankajsingh/workbench/Semester 5/Advance Programming Lab/Experiment_7/"nquees

The Placement of Queen in 5X5 Board as :

 1  0  0  0  0
 0  0  0  1  0
 0  1  0  0  0
 0  0  0  0  1
 0  0  1  0  0
→  Experiment_7
```

7. Learning Outcomes:

1. Learned the concepts of Backtracking algorithm
2. Learned the concepts to place n Queens
3. Learned to write a program for the above problem.
4. Learned to use VS code effectively.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

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