RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Lab Report 8

Course Code: *CSE 2202*

Course Title: *Sessional Based on CSE 2201.*

Submitted By:

Name : Shanjid Hasan Nishat

Roll No : 1803172

Section : 'C'

Date of Submission: 30/07/ 2021

Submitted To:

Biprodip Pal

Assistant Professor,

Dept. of Computer Science and Engineering.

Rajshahi University of Engineering & Technology

Problem Statement: Using backtracking, solve the N queens problem. For any N taken as input, your code should find out the goal nodes as well as the bounding nodes (from where no more nodes are checked along that path and backtracking occurred). Each state/node is represented by the following style.

Code:

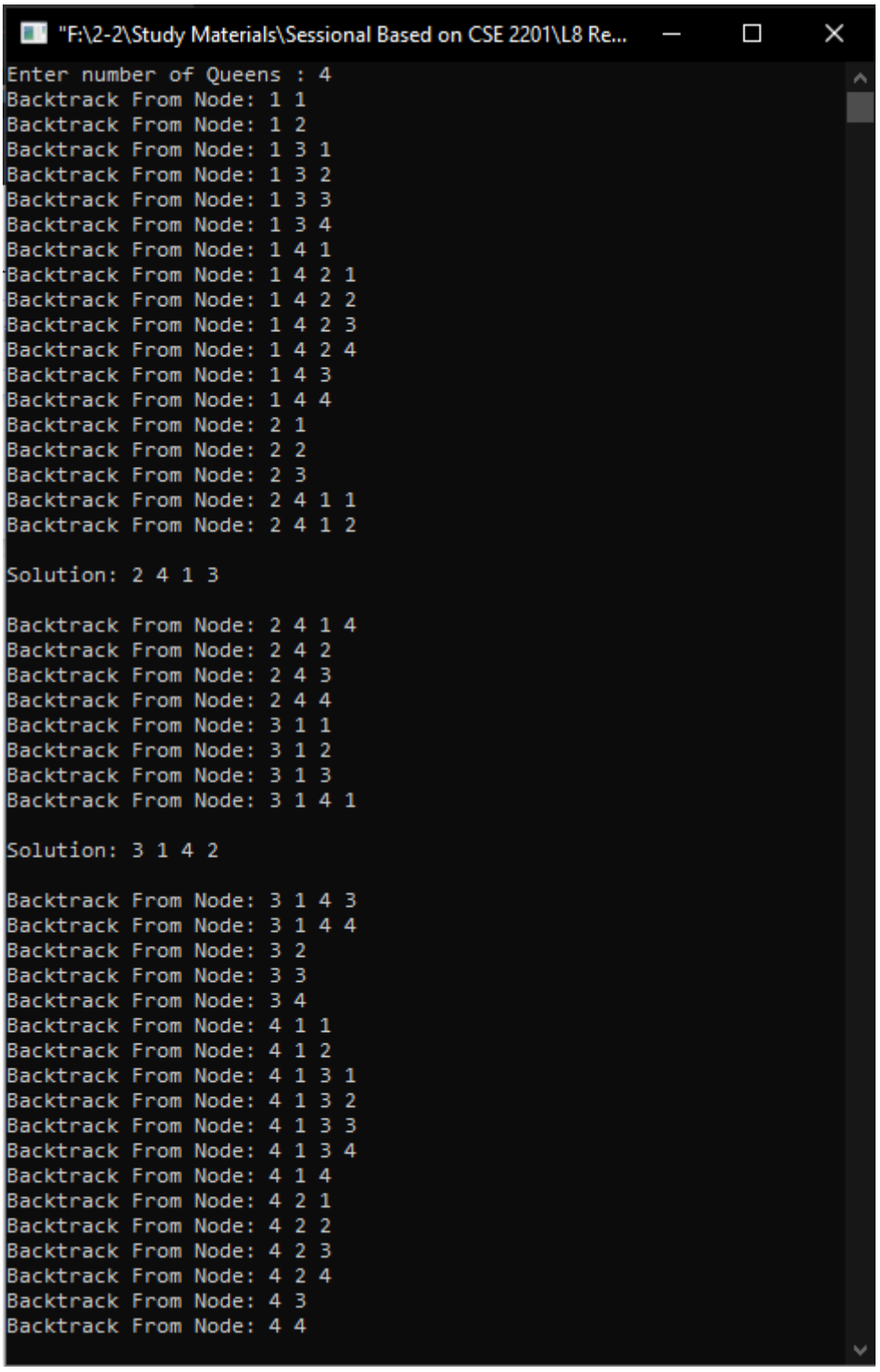
```
#include <bits/stdc++.h>
using namespace std;
bool isSafetoPlaceQueen(vector<vector<int>> board, int row, int col)
{
   int n = board.size();
   vector<int> nodes;
   for (int i = 0; i < n; i++)
   {
      for (int j = 0; j < n; j++)
      {
         if(i == row && j == col)
         {
            nodes.push_back(col+1);
         }
         if (board[i][j] == 1)
            nodes.push_back(j + 1);
      }
   }
   for (int i = 0; i < row; i++)
      if (board[i][col])
      {
         cout<<"Backtrack From Node: ";
         for(int j = 0 ; j < nodes.size() ; j++)
         {
            cout<<nodes[j]<<" ";
         }
         cout<<endl;
         return false;
```

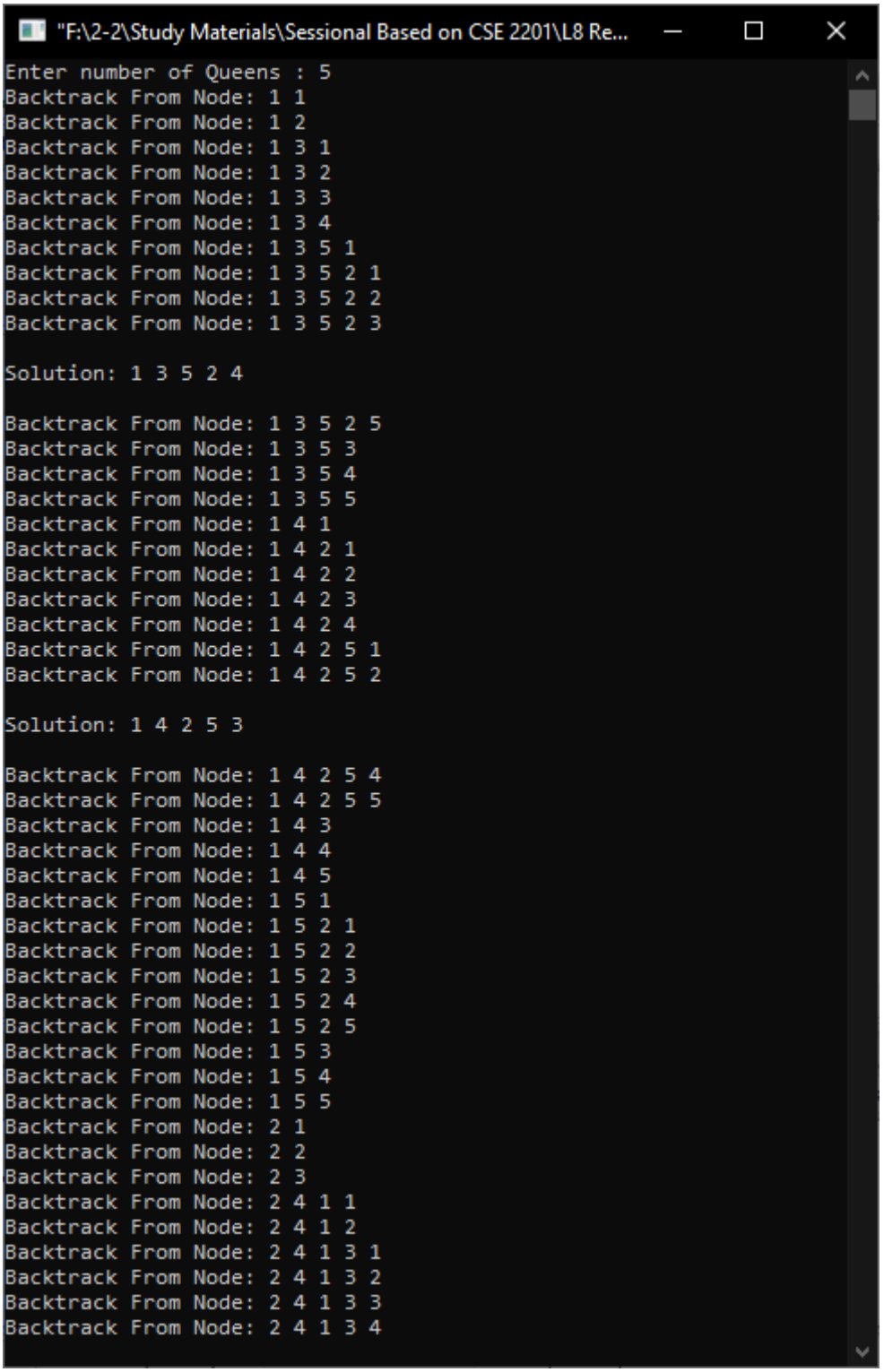
```
        }
    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j])
        {
            cout<<"Backtrack From Node: ";
            for(int j = 0 ; j < nodes.size() ; j++)
            {
                cout<<nodes[j]<<" ";
            }
            cout<<endl;
            return false;
        }
    for (int i = row, j = col; i >= 0 && j < n; i--, j++)
        if (board[i][j])
        {
            cout<<"Backtrack From Node: ";
            for(int j = 0 ; j < nodes.size() ; j++)
            {
                cout<<nodes[j]<<" ";
            }
            cout<<endl;
            return false;
        }

    return true;
}
void getSolution(vector<vector<int> >& board, int row)
{
    int n = board.size();
    if (row == n)
    {
        vector<int> nodes;
        for (int i = 0; i < n; i++)
        {
```

```
            for (int j = 0; j < n; j++)
            {
                if (board[i][j] == 1)
                    nodes.push_back(j + 1);
            }
        }
        cout<<endl<<"Solution: ";
        for(int i = 0 ; i < nodes.size() ; i++)
        {
            cout<<nodes[i]<<" ";
        }
        cout<<endl<<endl;
        return ;
    }
    for (int col = 0; col < n; col++)
    {
        if (isSafetoPlaceQueen(board, row, col))
        {
            board[row][col] = 1;
            getSolution(board, row + 1);
            board[row][col] = 0;
        }
    }
    return ;
}
int main()
{
    int n;
    cout<<"Enter number of Queens : ";
    cin>>n;
    vector<vector<int> > board(n, vector<int>(n, 0));
    getSolution(board, 0);
    return 0;
}
```

Output:


"F:\2-2\Study Materials\Sessional Based on CSE 2201\L8 Re...
Enter number of Queens : 4
Backtrack From Node: 1 1
Backtrack From Node: 1 2
Backtrack From Node: 1 3 1
Backtrack From Node: 1 3 2
Backtrack From Node: 1 3 3
Backtrack From Node: 1 3 4
Backtrack From Node: 1 4 1
Backtrack From Node: 1 4 2 1
Backtrack From Node: 1 4 2 2
Backtrack From Node: 1 4 2 3
Backtrack From Node: 1 4 2 4
Backtrack From Node: 1 4 3
Backtrack From Node: 1 4 4
Backtrack From Node: 2 1
Backtrack From Node: 2 2
Backtrack From Node: 2 3
Backtrack From Node: 2 4 1 1
Backtrack From Node: 2 4 1 2
Solution: 2 4 1 3
Backtrack From Node: 2 4 1 4
Backtrack From Node: 2 4 2
Backtrack From Node: 2 4 3
Backtrack From Node: 2 4 4
Backtrack From Node: 3 1 1
Backtrack From Node: 3 1 2
Backtrack From Node: 3 1 3
Backtrack From Node: 3 1 4 1
Solution: 3 1 4 2
Backtrack From Node: 3 1 4 3
Backtrack From Node: 3 1 4 4
Backtrack From Node: 3 2
Backtrack From Node: 3 3
Backtrack From Node: 3 4
Backtrack From Node: 4 1 1
Backtrack From Node: 4 1 2
Backtrack From Node: 4 1 3 1
Backtrack From Node: 4 1 3 2
Backtrack From Node: 4 1 3 3
Backtrack From Node: 4 1 3 4
Backtrack From Node: 4 1 4
Backtrack From Node: 4 2 1
Backtrack From Node: 4 2 2
Backtrack From Node: 4 2 3
Backtrack From Node: 4 2 4
Backtrack From Node: 4 3
Backtrack From Node: 4 4


"F:\2-2\Study Materials\Sessional Based on CSE 2201\L8 Re...
Enter number of Queens : 5
Backtrack From Node: 1 1
Backtrack From Node: 1 2
Backtrack From Node: 1 3 1
Backtrack From Node: 1 3 2
Backtrack From Node: 1 3 3
Backtrack From Node: 1 3 4
Backtrack From Node: 1 3 5 1
Backtrack From Node: 1 3 5 2 1
Backtrack From Node: 1 3 5 2 2
Backtrack From Node: 1 3 5 2 3
Solution: 1 3 5 2 4
Backtrack From Node: 1 3 5 2 5
Backtrack From Node: 1 3 5 3
Backtrack From Node: 1 3 5 4
Backtrack From Node: 1 3 5 5
Backtrack From Node: 1 4 1
Backtrack From Node: 1 4 2 1
Backtrack From Node: 1 4 2 2
Backtrack From Node: 1 4 2 3
Backtrack From Node: 1 4 2 4
Backtrack From Node: 1 4 2 5 1
Backtrack From Node: 1 4 2 5 2
Solution: 1 4 2 5 3
Backtrack From Node: 1 4 2 5 4
Backtrack From Node: 1 4 2 5 5
Backtrack From Node: 1 4 3
Backtrack From Node: 1 4 4
Backtrack From Node: 1 4 5
Backtrack From Node: 1 5 1
Backtrack From Node: 1 5 2 1
Backtrack From Node: 1 5 2 2
Backtrack From Node: 1 5 2 3
Backtrack From Node: 1 5 2 4
Backtrack From Node: 1 5 2 5
Backtrack From Node: 1 5 3
Backtrack From Node: 1 5 4
Backtrack From Node: 1 5 5
Backtrack From Node: 2 1
Backtrack From Node: 2 2
Backtrack From Node: 2 3
Backtrack From Node: 2 4 1 1
Backtrack From Node: 2 4 1 2
Backtrack From Node: 2 4 1 3 1
Backtrack From Node: 2 4 1 3 2
Backtrack From Node: 2 4 1 3 3
Backtrack From Node: 2 4 1 3 4