```
/*_____________________________________________________________________________

Assignment no :-4

Title:- Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and
Backtracking for n-queens problem or a graph coloring problem.

Name:-Pravin Jain        Roll No:-74        Batch:-T4        Subject:-AI

_____________________________________________________________________________*/

#include <iostream>

using namespace std;

int grid[10][10];

void print(int n) {
   for (int i = 0;i <= n-1; i++) {
      for (int j = 0;j <= n-1; j++) {


            cout <<grid[i][j]<< " ";


      }
      cout<<endl;
   }
   cout<<endl;
   cout<<endl;
}
bool isSafe(int col, int row, int n) {
  //check for same column
   for (int i = 0; i < row; i++) {
      if (grid[i][col]) {
         return false;
      }
   }
   //check for upper left diagonal
   for (int i = row,j = col;i >= 0 && j >= 0; i--,j--) {
      if (grid[i][j]) {
```

```
            return false;
        }
    }
    //check for upper right diagonal
    for (int i = row, j = col; i >= 0 && j < n; j++, i--) {
        if (grid[i][j]) {
            return false;
        }
    }
    return true;
}

bool solve (int n, int row) {
    if (n == row) {
        print(n);
        return true;
    }
    bool res = false;
    for (int i = 0;i <=n-1;i++) {
        if (isSafe(i, row, n)) {
            grid[row][i] = 1;
            res = solve(n, row+1) || res;
            grid[row][i] = 0;
        }
    }
    return res;
}
int main()
{
  ios_base::sync_with_stdio(false);
    cin.tie(NULL);
```

```
    int n;
    cout<<"Enter the number of queen"<<endl;
    cin >> n;
    for (int i = 0;i < n;i++) {
       for (int j = 0;j < n;j++) {
          grid[i][j] = 0;
       }
    }
    bool res = solve(n, 0);
    if(res == false) {
       cout << -1 << endl;
    } else {
       cout << endl;
    }
  return 0;
}

/*_____________________________________________________________________________
Output:-
Enter the number of queen
6
0 1 0 0 0 0
0 0 0 1 0 0
0 0 0 0 0 1
1 0 0 0 0 0
0 0 1 0 0 0
0 0 0 0 1 0
```

0 0 1 0 0 0

0 0 0 0 0 1

0 1 0 0 0 0

0 0 0 0 1 0

1 0 0 0 0 0

0 0 0 1 0 0

0 0 0 1 0 0

1 0 0 0 0 0

0 0 0 0 1 0

0 0 1 0 0 0

0 0 0 0 0 1

0 0 1 0 0 0

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

_____*/