DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

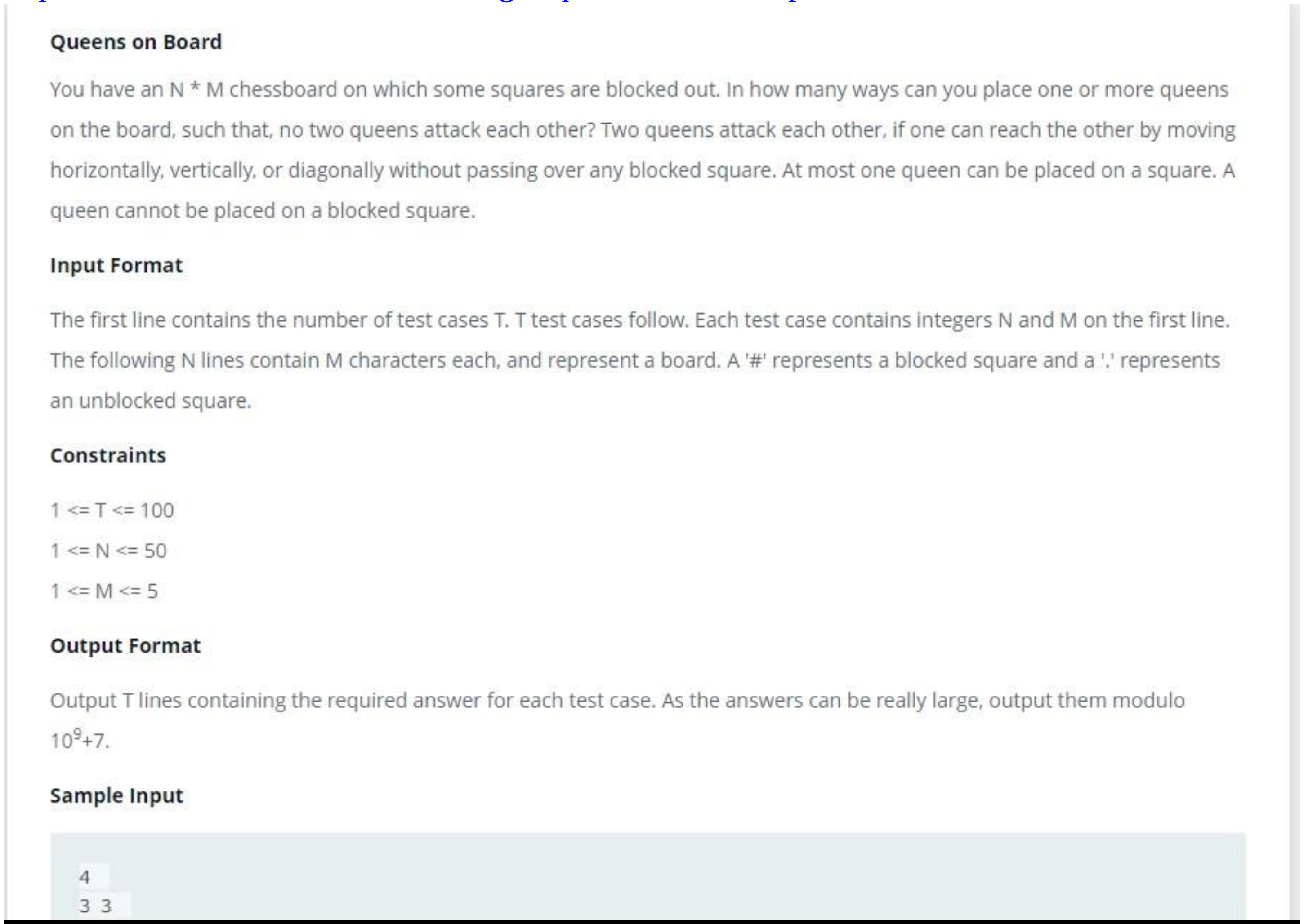
Experiment 9

Competitive Coding Lab (Backtracking)

Student Name: SANSKAR AGRAWAL **UID: 20BCS5914**
Branch: CSE **Section/Group: MM-806/B**
Semester: 5th **Date of Performance: 08/11/22**
Subject Name: Competitive Coding(CC) **Subject Code: 20CSP-314**

PROBLEM STATEMENT 9.1: -

https://www.hackerrank.com/challenges/queens-on-board/problem



Queens on Board

You have an N * M chessboard on which some squares are blocked out. In how many ways can you place one or more queens on the board, such that, no two queens attack each other? Two queens attack each other, if one can reach the other by moving horizontally, vertically, or diagonally without passing over any blocked square. At most one queen can be placed on a square. A queen cannot be placed on a blocked square.

Input Format

The first line contains the number of test cases T. T test cases follow. Each test case contains integers N and M on the first line. The following N lines contain M characters each, and represent a board. A '#' represents a blocked square and a '.' represents an unblocked square.

Constraints

1 <= T <= 100

1 <= N <= 50

1 <= M <= 5

Output Format

Output T lines containing the required answer for each test case. As the answers can be really large, output them modulo 10^9+7.

Sample Input

```
4
3 3
```

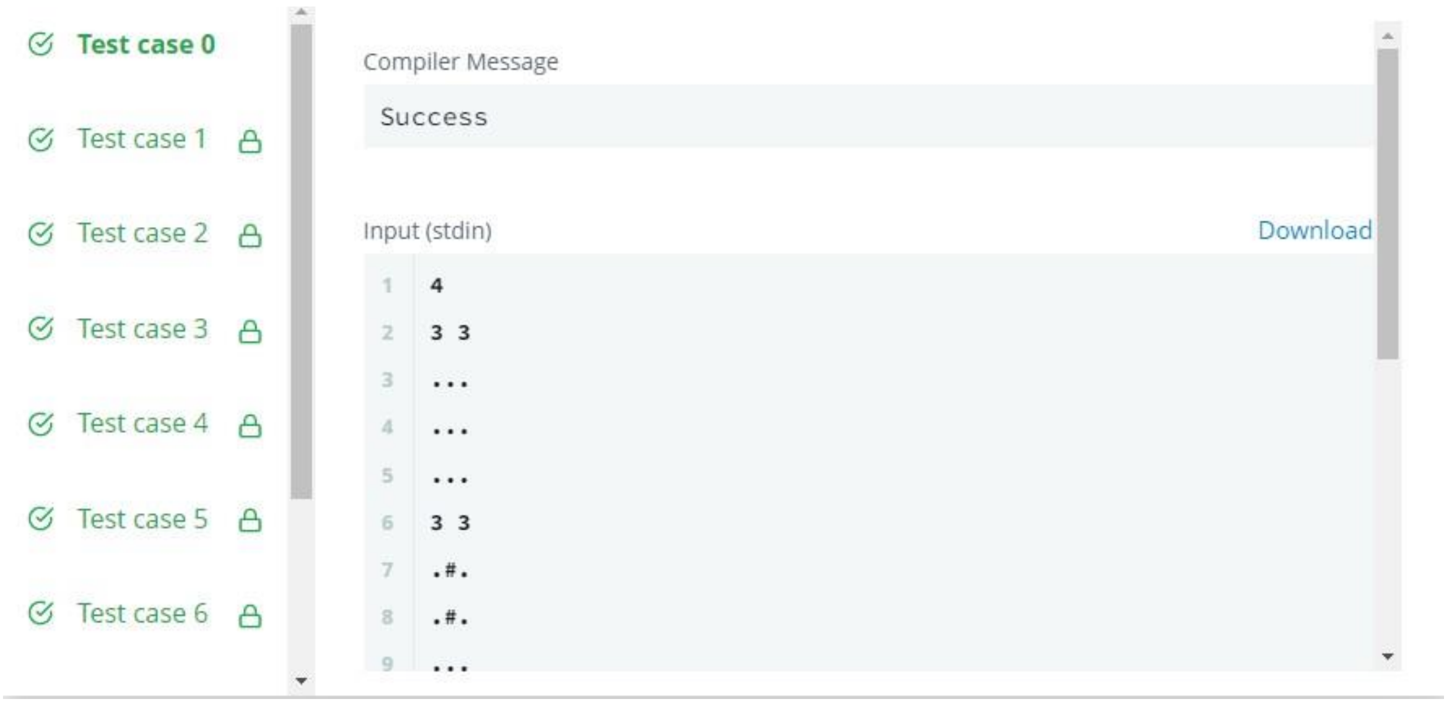
SOLUTION:

```
  static String createRowConfig(int mask)
  {
    String rowConfig = Integer.toString(mask, 2);

    while(rowConfig.length() < m) {
      rowConfig = "0" + rowConfig;
    }
    return rowConfig;
  }


  static boolean isValidRowConfig(String rowConf, int row) {
int count = 0;

    for(int i = 0; i < m; i++) {       if
(board[row].charAt(i) == '#') {
if (hasQueen(rowConf, i)) {
          return false;
        }

        count = 0;
continue;
      }
      if (hasQueen(rowConf, i)) {
if (++count > 1) {           return
false;
        }
      }     }
return true;
  }
```

```
  static boolean compatible(String rowConf, String attVect) {
    for(int i = 0; i < m; i++) {
      if (!hasQueen(rowConf, i))
      {
continue;
      }
      if (attackedFromUpperLeft(i, attVect) || attackedFromAbove(i, attVect) ||
attackedFromUpperRight(i, attVect))
      {
        return false;
      }     }
return true;
  }

  static boolean isOpenSpace(int row, int col) {
    if (row < 0 || row >= n) {
      return false;
    }
    if (col < 0 || col >= m) {
      return false;
    }
    return board[row].charAt(col) != '#';
  }

  static boolean hasQueen(String rowConf, int col) {
    if (col < 0 || col >= m) {
      return false;
    }
    return rowConf.charAt(col) == '1';
  }

  static boolean attackedFromUpperLeft(int col, String attVect) {
if (col <= 0) {       return false;
    }
    return attVect.charAt(col * 3) == '1';
  }



  static boolean attackedFromAbove(int col, String attVect) {
return attVect.charAt((col * 3) + 1) == '1';   }

  static boolean attackedFromUpperRight(int col, String attVect) {
if (col >= m - 1) {       return false;
```

```
    }
    return attVect.charAt((col * 3) + 2) == '1';
}
```

TEST CASES:

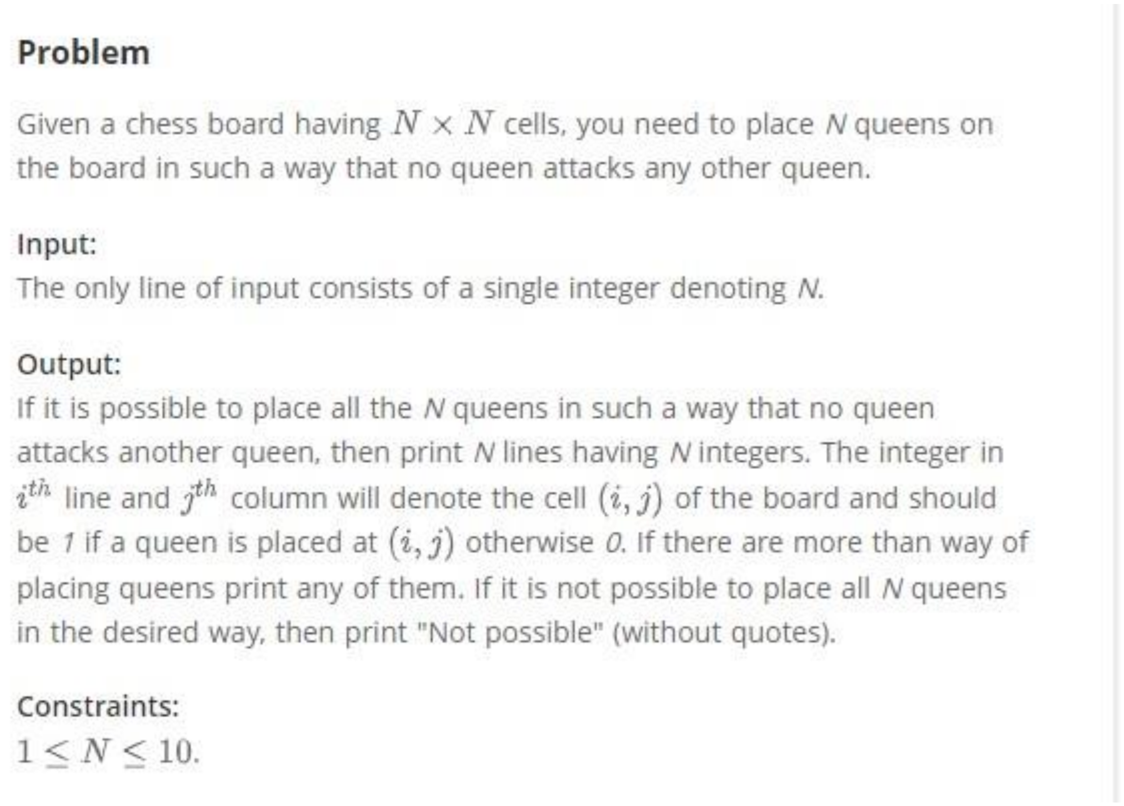


```
Success
```

```
4
3 3
...
...
...
3 3
.#.
.#.
...
```

PROBLEM STATEMENT 9.2: -

https://www.hackerearth.com/practice/basic-programming/recursion/recursion-andbacktracking/practice-problems/algorithm/n-queensrecursion-tutorial/



Problem

Given a chess board having $N \times N$ cells, you need to place *N* queens on the board in such a way that no queen attacks any other queen.

Input:
The only line of input consists of a single integer denoting *N*.

Output:
If it is possible to place all the *N* queens in such a way that no queen attacks another queen, then print *N* lines having *N* integers. The integer in i^{th} line and j^{th} column will denote the cell (i, j) of the board and should be *1* if a queen is placed at (i, j) otherwise *0*. If there are more than way of placing queens print any of them. If it is not possible to place all *N* queens in the desired way, then print "Not possible" (without quotes).

Constraints:
$1 \le N \le 10$.

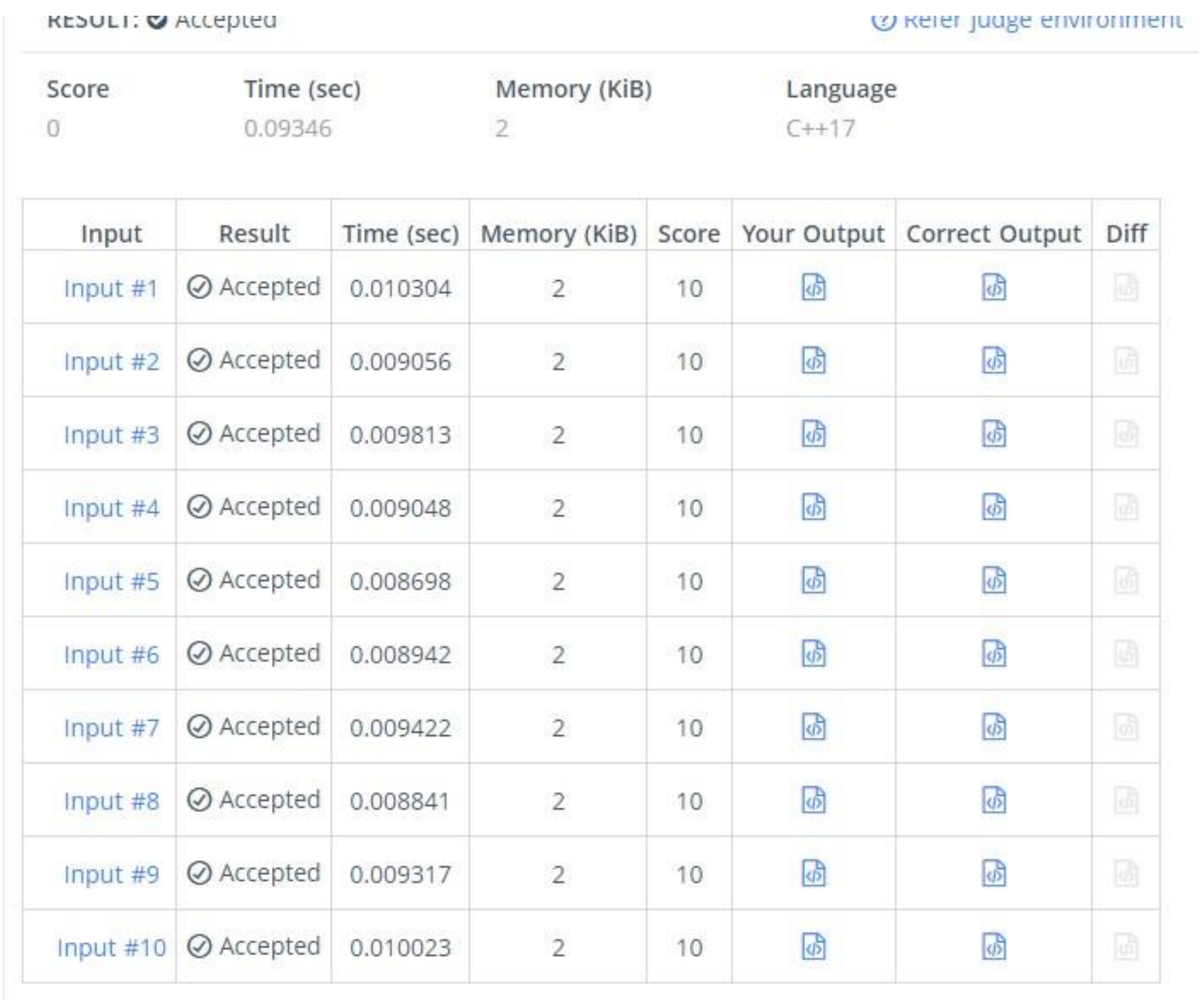
SOLUTION:

```
#include<iostream> using

namespace std;
```

```
bool issafe(int ** arr, int x, int y, int n)
{   for (int row = 0; row < x; row++) {
if (arr[row][y] == 1) {       return 0;
    }
}
  int row = x;
int col = y;
  while (row >= 0 && col >= 0) {
    if (arr[row][col] == 1)
{       return 0;     }
    row--;
col--;
  }
  row = x;
col = y;
  while (row >= 0 && col < n) {
    if (arr[row][col] == 1) {
return 0;
    }
    row--;
col++;   }
return 1;
}
```

```
bool nqueen(int ** arr, int x, int n)

{   if (x >= n) {     return 1;

  }


  for (int col = 0; col < n; col++)

{     if (issafe(arr, x, col, n)) {

arr[x][col] = 1;       if (nqueen(arr,

x + 1, n)) {         return 1;       }

      arr[x][col] = 0;

    }   }

return 0;

}


int main() {
  int n;

  cin >> n;   int ** arr =

new int * [n];   for (int i =

0; i < n; i++) {     arr[i] =

new int[n];     for (int j = 0;

j < n; j++) {       arr[i][j] =

0;

    }

  }
  if (nqueen(arr, 0, n)) {
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
cout << arr[i][j] << " ";
      }
      cout << endl;
    }   }
else {
    cout << "Not possible" << endl;
  }
}
```

TEST CASES:



RESULT: Accepted   Refer judge environment

Score	Time (sec)	Memory (KiB)	Language
0	0.09346	2	C++17

Input	Result	Time (sec)	Memory (KiB)	Score	Your Output	Correct Output	Diff
Input #1	Accepted	0.010304	2	10			
Input #2	Accepted	0.009056	2	10			
Input #3	Accepted	0.009813	2	10			
Input #4	Accepted	0.009048	2	10			
Input #5	Accepted	0.008698	2	10			
Input #6	Accepted	0.008942	2	10			
Input #7	Accepted	0.009422	2	10			
Input #8	Accepted	0.008841	2	10			
Input #9	Accepted	0.009317	2	10			
Input #10	Accepted	0.010023	2	10			