## CHANDIGARH UNIVERSITY

## UNIVERSITY INSTITUTE OF NGINEERING

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By: Submitted To:**  Vivek Kumar(21BCS8129) Mamta Punia(E12337) | |
| **Subject Name** | Competitive Coding - I |
| **Subject Code** | 20CSP-314 |
| **Branch** | Computer Science and Engineering |
| **Semester** | 5th |

**Experiment No. - 9**

**Student Name: Vivek Kumar UID: 21BCS8129**

**Branch: BE-CSE(LEET) Section/Group: WM-20BCS-616/A**

**Semester: 5th Date of Performance: 4/11/2022**

**Subject Name: Competitive coding - I Subject Code: 20CSP-314**

# **Construct the Array**

1. **Aim/Overview of the practical:**

Backtracking

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.

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

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

1. **Objective:**
   * To understand the concept of Backtracking.

**4. Code:**

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

class Solution {

static int NS = 50;

static int MS = 5;

static int K = 32;

static int[][][][]s = new int[NS][K][K][K];

static int AK;

static int[][] mp= new int[NS][MS];

static int n;

static int m;

static int dp(int c, int b1, int b2, int b3) {

if (c == n) return 1;

if (s[c][b1][b2][b3] >= 0) return s[c][b1][b2][b3];

/\*

System.out.print(c);System.out.print(' ');

System.out.print(b1);System.out.print(' ');

System.out.print(b2);System.out.print(' ');

System.out.print(b3);System.out.print(' ');

System.out.println();

\*/

int sum = 0;

for (int i = 0; i < AK; i++) {

if (check(c,i,b1,b2,b3)){

int[] mask = mask(c,i,b1,b2,b3);

/\*

System.out.print(c);

System.out.println(i);

\*/

sum = (sum+dp(c+1, mask[0],mask[1],mask[2]))%1000000007;

}

}

s[c][b1][b2][b3] = sum;

return sum;

}

static boolean check(int c, int i, int b1, int b2, int b3) {

int[] loc = {1,2,4,8,16};

boolean selfblock = false;

//check other block

for (int li = 0; li < m; li++) {

if ((i&loc[li]) != 0) {

if (mp[c][li] == 0) return false;

if (selfblock == true) return false;

if ((b1&loc[li]) !=0 || (b2&loc[li]) !=0|| (b3&loc[li]) !=0) return false;

selfblock = true;

}

if (mp[c][li] == 0) selfblock = false;

}

return true;

}

static int[] mask(int c, int i, int b1, int b2, int b3){

int[] loc = {1,2,4,8,16};

int[] mask = new int[3];

mask[0] = b1|i;

mask[1] = ((b2 << 1) % K) | ((i << 1) % K);

mask[2] = (b3 >> 1) | (i >> 1);

for (int li = 0; li < m; li++){

if(mp[c][li] == 0) {

mask[0] = mask[0] & (~loc[li]);

mask[1] = mask[1] & (~(loc[li] << 1)%K);

mask[2] = mask[2] & (~(loc[li] >> 1));

}

}

return mask;

}

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int TN = in.nextInt();

for (int ti = 0; ti < TN; ti++) {

n = in.nextInt();

m = in.nextInt();

String str[] = new String[n];

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

str[i] = in.next();

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

for (int j = 0; j < m; j++) {

mp[i][j] = 1;

if (str[i].charAt(j) == '#')

mp[i][j] = 0;

}

for (int i1 = 0; i1 < n; i1++)

for (int i2 = 0; i2 < K; i2++)

for (int i3 = 0; i3 < K; i3++)

for (int i4 = 0; i4 < K; i4++)

s[i1][i2][i3][i4] = -1;

AK = 1;

for (int i =0; i< m; i++)

AK = AK \*2;

int r = dp(0,0,0,0);

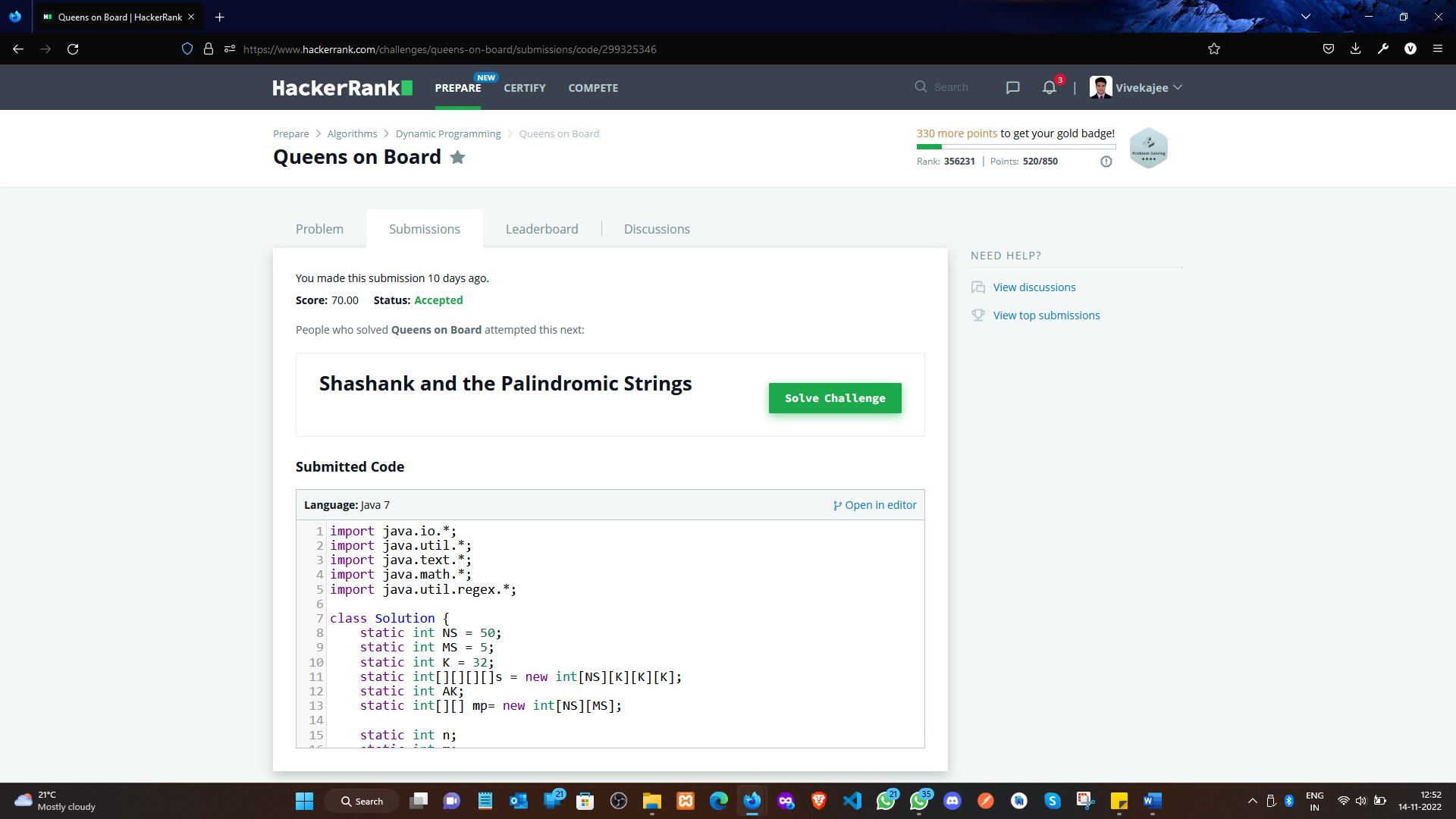
System.out.println((r+1000000007-1) % 1000000007);

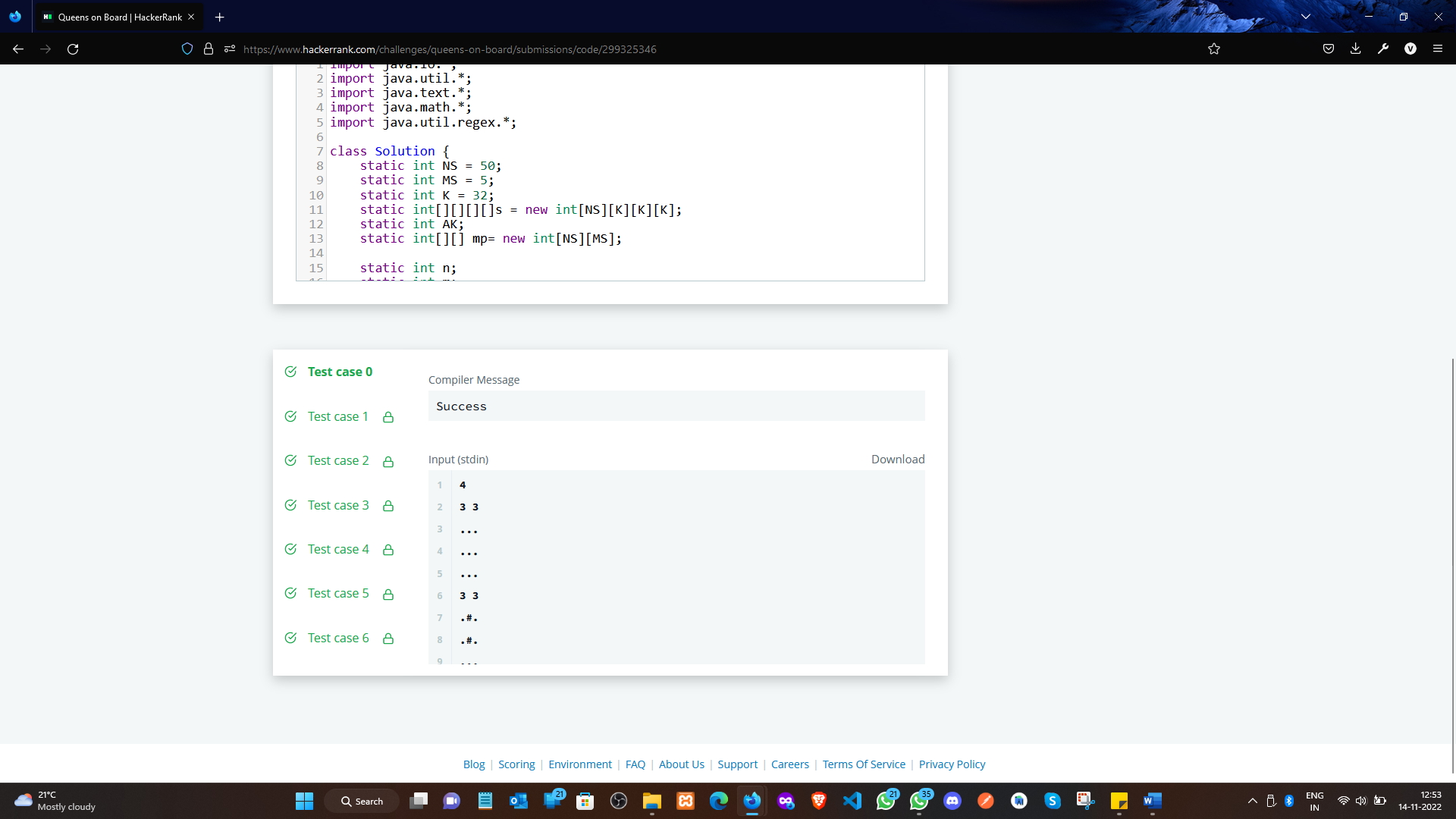
}

}

}

**5. Result/Output/Writing Summary:**





**Experiment 9.2**

1. **Aim/Overview of the practical:**

Backtracking

You are given a list of N positive integers, A = {a[1], a[2], ..., a[N]} and another integer S. You have to find whether there exists a non-empty subset of A whose sum is greater than or equal to S.

You have to print the size of minimal subset whose sum is greater than or equal to S. If there exists no such subset then print -1 instead.

<https://www.hackerrank.com/challenges/subset-sum/problem>

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

1. **Objective:**
   * To understand the concept of Backtracking.
2. **Code:**

import java.util.Scanner

import scala.collection.Searching.\_

object Solution {

  def main(args: Array[String]): Unit = {

    val sc = new Scanner(System.in)

    sc.nextLine

    val a = sc.nextLine.split(' ').map(\_.toLong).sortBy(-\_)

    var sum = 0L

    val sums = a.map(v => {

      sum += v

      sum

    })

    val t = sc.nextInt

    (0 until t).foreach(\_ => {

      val s = sc.nextLong

      val count = (sums.search(s) match {

        case InsertionPoint(i) => i

        case Found(i) => i

      }) + 1

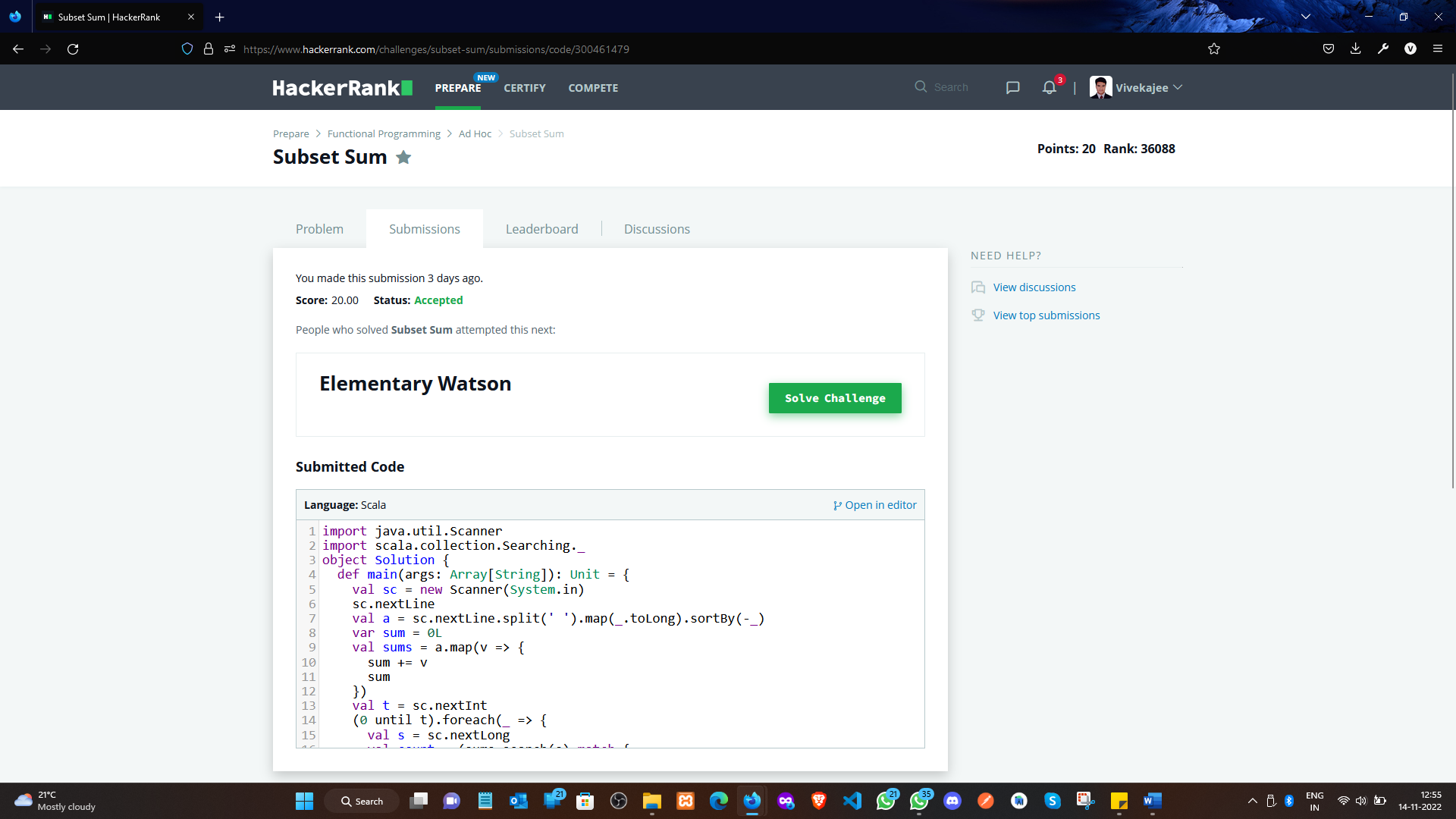
      println(if (count <= a.length) count else -1)

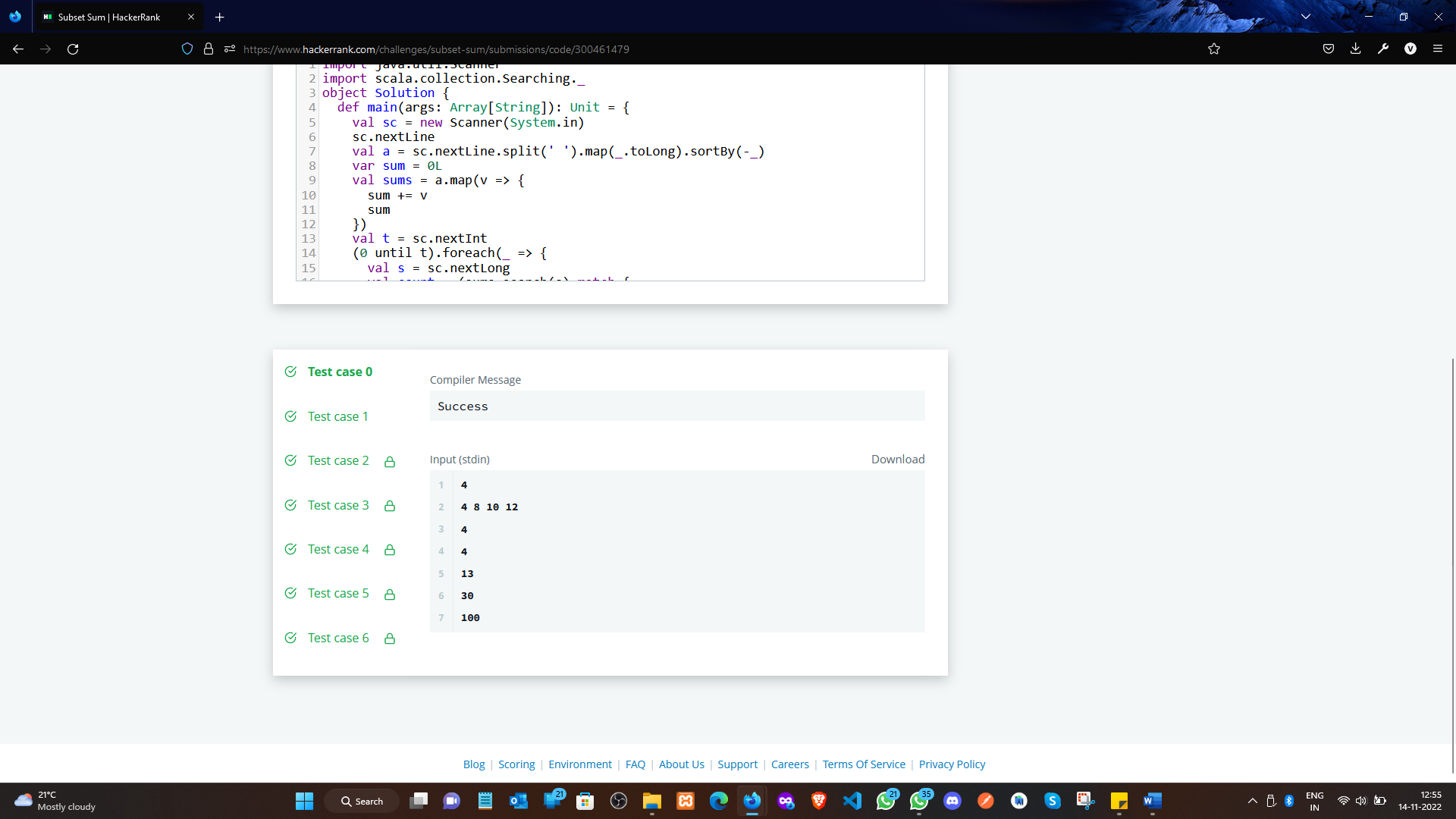
    })

  }

}

1. **Result/Output/Writing Summary:**





**Learning outcomes (What I have learnt):**

* + Learned the concept of Backtracking.

**Evaluation Grid (To be created per the faculty's SOP and Assessment guidelines):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. | Worksheet completion including writing learning objectives/Outcomes.  (To be submitted at the end of the day). |  |  |
| 2. | Post-Lab Quiz Result. |  |  |
| 3. | Student Engagement in  Simulation/Demonstration/Performance and Controls/Pre-Lab Questions. |  |  |
|  | Signature of Faculty (with Date): | Total Marks Obtained: |  |