



Medisetti S R V Lakshmi Narendra

Candidate First Name: Medisetti
(As per Passport/Aadhar)

Candidate Last Name: Sri Ranga Veera Lakshmi Narendra
(As per Passport/Aadhar)

Notice Period of the Candidate: NA

Internal or External Candidate: Internal

Interview Availability Date: Immediate

Start Availability Date: Immediate



Has the candidate worked directly for Ford before? (Yes/No) No

Has the candidate worked for Ford as an Agency worker before? (Yes/No) No

If yes to either of the above, please specify the below details:

- i. CDSID:
- ii. Supervisor:
- iii. Duration of the Project:
- iv. Exit reason:

Overall IT Experience: 3.5 years

Experience in Core Skill: 3.5 years

Other relevant Qualification/Certifications, if any:

Hacker rank Score: 237 / 300

Candidate's Top 3 relevant Skills:

- i. Fullstack Development (Java 1.8 , Spring Boot 2.2 , Angular 8)
- ii. Development Tools:
 1. IDEs (Eclipse 4.4, STS 4, Visual Studio Code 1.1, IntelliJ Idea)
 2. Build Tools (Maven 3.3, Gradle, GitHub)
- iii. RDBMS (MySQL)



EDUCATION HISTORY

Educational History:

University Degree Bachelor

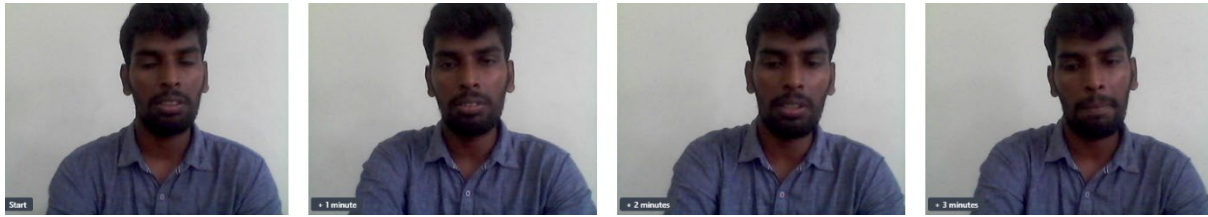
06.2013 – 05.2017

Bachelors of Technology,
Electronics and Communication Engineering,
Jawaharlal Nehru Technological University Kakinada (JNTUK)

EMPLOYMENT HISTORY (Please start with most recent employment)

* Please find the below attached profile

MEDISETTI SRI RANGA VEERA LAKSHMINARENDRA



PROFESSIONAL SUMMARY

- Skilled Java full stack developer with good experience in development environment
- Strong hands on experience in Java, Spring Boot and Rest API
- Strong experience in Front-end development using JavaScript, Angular, HTML and CSS
- Good knowledge of JUnit and Mockito.
- Good knowledge in Software Development Lifecycle and having Good communication skills
- Adaptable to learn new skills and technologies quickly for enhancing productivity
- Team player with utmost dedication towards profession

EDUCATION

Degree	University
Bachelor Of Technology(ECE)	JNTUK University

EXPERIENCE

Organization	Designation	Duration
STG Infotech	Software Engineer	March 2020 to till Date

TECHNICAL SKILLS

PROGRAMMING LANGUAGES	Java 1.8
FRAMEWORKS	Spring (Core ,MVC, Boot) 5.X
WEB TECHNOLOGIES	Servlets, XML, JSON, REST Webservices
TESTING	JUnit 5, Mockito.
FRONT END TOOLS	Angular 12, HTML 5 ,CSS 3, JavaScript, Bootstrap 5
BACK END TOOLS	Node.js
IDE'S & TOOLS	Eclipse, STS, Visual Studio Code
WEB SERVERS	Tomcat 8
DATABASE/RDBMS	MySQL
BUILD TOOLS	Maven 3.3,GitHub

PROJECTS PROFILE

Project Name	Personal Car Finance
Client	Leading Global Automotive Financing Institution
Organization	STG Infotech
Role	Full Stack Developer
Tools Used	Java 1.8, Spring Boot, Angular 12, MySQL 8.0

Project Description

Personal Car Finance application allows a user to manage their vehicle finance details on the go and it comes loaded with features designed for speed and simplicity. Just a few of the many app features:

- Register Your Account
- Biometric or PIN login
- Make Payments including Principal Only & Recurring
- View/Edit/Delete Scheduled Payments
- Enroll in Auto Pay
- **View/Edit Profile**
- **View Transaction History**
- **View Statements**
- **View Contract Information**
- Document Center
- Vehicle Mileage Information & Reporting

Responsibilities

- Understand client requirements
- Coding and unit testing (CUT) the above-mentioned highlighted features, releasing the tested source code for QA build
- Understand quality assurance team reported defects and fixed the same within the test sprint.
- Collaborating with DevOps team in releasing the build for QA and UAT environments

Project Name	LTS Accounting Servicing
Client	Leading Bank Institution
Organization	STG Infotech
Role	Devloper
Tools Used	Java 1.8, Spring Boot, Angular 12, MySQL 8.0

Project Description

LTS Accounting Servicing application is owned by the Lending Trade Services Department also known as Loan Maintenance Portal that allows a user to raise a request based on their requirement like Address Changes, Biling Schedule Changes, payment corrections, fees for maintenance of the user credits and other issues related to the user Loan details designed to load in inferior time. This has approval process, Admin & Analyst can approve the request that are created by the Preparer. Some of the app features are shown below

- Creating a Maintenance Request.
- User Profile to maintain all the roles of the employees
- User Roles to maintain all the permissions of the user (Security Metrix)

- View/Edit/Delete Users Request
- View/Edit User Profile
- Multilevel Approvals
- Exporting data to Excel
- View Requests Information
- Maintaining Document
- Users Requests Information & Reporting

Responsibilities

- Understand client requirements
- Coding and unit testing (CUT) the above-mentioned highlighted features, releasing the tested source code for QA build
- Understand quality assurance team reported defects and fixed the same within the test sprint
- Collaborating with DevOps team in releasing the build for QA and UAT environments



You can view this report online at : <https://www.hackerrank.com/x/tests/1045899/candidates/41465497/report>

Full Name:	mediseti sri ranga veera lakshminarendra
Email:	narendramedisetti423@gmail.com
Test Name:	Software Engineer Skills assessment - STG Infotech India LLP
Taken On:	23 Jun 2022 16:07:24 IST
Time Taken:	111 min 19 sec/ 120 min
Work Experience:	1 years
Why are you taking this challenge?:	Candidate Evaluation for Submission
Invited by:	Venkat
Invited on:	16 Jun 2022 10:46:42 IST
Skills Score:	<div>C# (Intermediate) 50/50</div> <div>C++ (Advanced) 50/50</div> <div>Java (Basic) 50/50</div> <div>Problem Solving (Basic) 50/50</div> <div>Problem Solving (Intermediate) 87/150</div> <div>SQL (Basic) 50/50</div>
Tags Score:	<div>Algorithms 137/200</div> <div>Arrays 12/75</div> <div>C# 50/50</div> <div>C++ 50/50</div> <div>Combinatorics 75/75</div> <div>Data Structures 12/75</div> <div>Databases 50/50</div> <div>Dynamic Programming 12/75</div> <div>Easy 150/150</div> <div>Implementation 50/50</div> <div>Interviewer Guidelines 187/250</div> <div>Java 50/50</div> <div>Medium 87/150</div> <div>OOPS 50/50</div> <div>Problem Solving 137/200</div> <div>SQL 50/50</div> <div>Simple Joins 50/50</div>

79%

237/300

scored in **Software Engineer Skills assessment - STG**
Infotech India LLP in 111 min
19 sec on 23 Jun 2022 16:07:24 IST

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

	Question Description	Time Taken	Score	Status
Q1	Double on Match > Coding	6 min 37 sec	50/ 50	✔
Q2	Team Formation 2 > Coding	34 min 3 sec	75/ 75	✔
Q3	Distance Between Two Points > Coding	13 min 25 sec	50/ 50	⚠
Q4	Scheduling Errors > DbRank	5 min 26 sec	50/ 50	✔
Q5	Sprint Training > Coding	45 min 54 sec	12/ 75	⚠

QUESTION 1

✔

Correct Answer

Score 50

Double on Match > Coding

EasyAlgorithmsProblem SolvingInterviewer Guidelines

QUESTION DESCRIPTION

Given an array of long integers '*arr*' and a number '*num*', iterate through the elements in *arr* and double the value of *num* whenever an element equals *num*. Find the maximum possible value of *num*, knowing that *arr* can be reordered before the iteration.

Example

arr = [1, 2, 4, 11, 12, 8]

num = 2

Iterating through *arr*:

	<i>arr</i>	<i>num</i>
	2	2
1	2	
2	4	
4	8	
11	8	
12	8	
8	16	

The maximal value of *num* = 16. Note that *arr* could have been reordered before iterating.

Function Description

Complete the function *doubleSize* in the editor below.

doubleSize has the following parameter(s):

- long int arr[n]*: an array of long integers
- long int num*: the base long integer

Returns:

- long int*: the maximal value of *num*

Constraints

- $1 \leq n \leq 10^6$
- $0 \leq arr[i] \leq 10^{16}$
- $0 \leq num \leq 10^4$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n , the size of the array arr .

Each of the next n lines contains an integer $arr[i]$ where $0 \leq i < n$.

The last line contains a long integer, num .

▼ Sample Case 0

Sample Input 0

STDIN	Function
5	→ arr[] size n = 5
1	→ arr = [1, 2, 3, 1, 2]
2	
3	
1	
2	
1	→ num = 1

Sample Output 0

4

Explanation 0

Rearrange arr to $arr = \{1, 1, 2, 2, 3\}$.

arr	num
	1
1	2
1	2
2	4
2	4
3	4

▼ Sample Case 1

Sample Input 1

STDIN	Function
3	→ arr[] size n = 3
1	→ arr = [1, 1, 1]
1	
1	
1	→ num = 1

Sample Output 1

2

Explanation 1

arr	num
	1
1	2
1	2
1	2

▼ Sample Case 2

Sample Input 2

```
STDIN      Function
-----
5   →      arr[] size n = 5
2   →      arr = [2, 5, 4, 6, 8]
5
4
6
8
2   →      num = 2
```

Sample Output 2

```
16
```

Explanation 2

Rearrange *arr* to *arr* = {2, 4, 5, 6, 8}.

arr	num
	2
2	4
4	8
5	8
6	8
8	16

INTERVIEWER GUIDELINES

▼ Hint 1

Imagine that the optimal ordering of the array results in doubling the value of *b* some certain number of times, for example, it doubles it 5 times. Think about what exact values in the array double the value of *b* while iterating over the array.

Answer: Those values will be *b*, *b**2, *b**4, *b**8, *b**16 and so on.

▼ Hint 2

Is there any ordering of the array that guarantees that the values *b*, *b**2, *b**4, *b**16, ... are processed in this exact order so we have as many doubles as possible?

Answer: Yes, ordering the array in a non-descending order guarantees that.

▼ Solution

Concepts Covered:

greedy algorithms

sorting

proving the correctness of a greedy approach

Optimal Solution:

The optimal strategy is to sort the array in non-descending order, which means from smallest to largest value. Iterate through the array in that order doubling the value of *b* accordingly. Let's call a match the situation when the current value of *b* equals some element of the array while iterating over the array. There are a few crucial observations one can make in order to justify that strategy:

First, assume that *b* > 0 because if *b* = 0, then any order of the array results in the final value of *b* equal 0.

1. The final value of b is determined solely by the number of matches while iterating over the array. It means that if there are k matches, then the final value is the initial value of b times 2^k . The more matches we have, the larger the final b is, so the task is reduced to the task of finding the order of the array that maximizes the number of matches.

2. The initial value of b is either 0 or a positive number. If it's 0, then it never changes, even when doubled, and when it's a positive number, it only increases, more precisely doubles, with every match.

Now, let's assume that the optimal solution has k matches. If $k = 0$, then the order in which we iterate over the array doesn't matter, so let's assume that $k > 0$. Then, the sequence of consecutive values denoting matches while iterating over the array must be: $b, b*2, b*4, \dots, b*2^{k-1}$ which is strictly increasing so if we order the array in non-descending order, we are guaranteed to cover all matches in the optimal solution.

We process the array in the sorted order and double b with every match. After the iteration, we return the final value of b .

Python Code:

```
def doubleSize(arr, b):
    arr.sort()
    for x in arr:
        if x == b:
            b *= 2
    return b
```

Additional discussion:

Is there any value of b for which the problem can be solved in constant time regardless of the array size and its content? **Answer:** yes, for $b=0$, the answer is always 0 regardless of the array.

Is integer overflow an issue considering the given constraints and why? **Answer:** no, because the final value of b won't exceed two times the maximum value of elements in the array which are at most 10^{16} .

Is it possible to discard some elements of b at the very beginning? If yes, which ones? **Answer:** yes, all elements smaller than b , all elements not divisible by b , in general, all elements that are not of the form $b*2^k$ for $k \geq 0$.

Assume that the elements of the array can be negative, but b stays non-negative. Does the solution change in that case? **Answer:** no, because in that case, all negative elements will be smaller than b , which means they can be ignored.

Assume that both the elements of the array and the value of b can be negative. Does the solution change in that case? **Answer:** yes, because if b is negative then the goal of maximizing the final value of b will be achieved if b is doubled the least possible times, not the most.

▼ Complexity Analysis

Time Complexity - $O(n * \log(n))$

The time complexity is dominated by the sorting phase which is $O(n * \log(n))$. After the array is sorted, we only iterate over it once. It takes constant time in each iteration, thus the total time complexity is $O(n * \log(n))$.

Space Complexity - $O(1)$ or $O(n)$

Depending on the underlying sorting algorithm, the space complexity can be either $O(1)$ or $O(n)$.

▼ Follow up Question 1

Can you find the final value of b faster than $O(n * \log(n))$? Assuming it can be done, are you able to produce the exact order in which the array should be iterated? Discuss the space complexity in both cases. How much can you reduce the space complexity?

a) Yes, it can be done faster. For example, with a hash set. The idea is to put all elements into the hash set and then find the longest sequence of form $b, b*2, b*4, b*8, \dots$ that exists in the hash set. It's important to point out that the time complexity of that solution is $O(n)$ but it is the expected time, not deterministic. There might be different implementations of that method, all based on the idea of finding

the longest sequence of form b , $b*2$, $b*4$, $b*8$, ... that exists in the array.

Python Code:

```
def doubleSize(arr, b):
    mem = set(arr)
    while b in mem:
        b *= 2
    return b
```

The space complexity of the above is $O(n)$.

b) It is also possible to produce the exact order in which the array should be processed in that case. The pseudo-code below illustrates how it can be done:

Python Code:

```
def doubleSize(arr, b):
    mem = set(arr)

    order = []
    while b in mem:
        order.append(b)
        b *= 2

    for x in arr:
        if x in mem:
            mem.remove(x)
        else:
            order.append(x)

    # order is the order in which the array should be
    # processed to produce the final b
    return b
```

c) Space complexity can be reduced further. One possible optimization is based on the observation that if $b > 0$, then the sequence of values b , $b*2$, $b*4$, $b*8$, ..., existing in the array won't be long, in fact, it won't have more than 60 elements because the maximum value in the array is 10^{16} . Then, one can declare an array of size 60 such that at a position with index i , it stores the information if value $b*2^i$ exists in the array.

Python Code:

```
def doubleSize(arr, b):
    exponent = {} # exponent[b*2^i] = i
    for e in range(61):
        exponent[b*2**e] = e
        e += 1

    mem = [False for _ in range(61)] # mem[i] = True iff b*2^i exists in
the array
    for x in arr:
        e = exponent.get(x)
        if e is not None:
            mem[e] = True

    e = 0
    while mem[e]:
        b *= 2
        e += 1
    return b
```

Imagine the original task, but you want b to be minimal at the end of the process. How would you approach that?

There are two cases to consider:

1. There is no value b in the array:

In this case, no matter in what order the array is iterated over, the initial value of b will never get doubled, so its final value will be equal to its initial value b and this is the minimum that can be achieved.

2. There is at least one value b in the array:

In this case, we can always achieve the value of $b*2$ at the end of iteration by placing all values equal to b at the end of the array, and all different values at the front of the array. By doing that, we're guaranteed to double the initial value of b only once. Moreover, it's not possible to achieve value smaller than $2*b$ because at least one b exists in the array and it will be met at some point of iteration.

Python Code:

```
def doubleSize(arr, b):
    for x in arr:
        if x == b:
            return 2*b
    return b
```

CANDIDATE ANSWER

Language used: **Java 8**

```
1  class Result {
2
3      /*
4       * Complete the 'doubleSize' function below.
5       *
6       * The function is expected to return a LONG_INTEGER.
7       * The function accepts following parameters:
8       * 1. LONG_INTEGER_ARRAY arr
9       * 2. LONG_INTEGER b
10      */
11
12     public static long doubleSize(List<Long> arr_44, long b_44) {
13         Collections.sort(arr_44);
14         for (int i = 0; i < arr_44.size(); i++) {
15             if(arr_44.get(i)==b_44) {
16                 b_44=b_44*2;
17             }
18         }
19         return b_44;
20     }
21 }
22
23 }
24
25 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
----------	------------	------	--------	-------	------------	-------------

TestCase 0	Easy	Sample case	✔	Success	1	0.1885 sec	30.1 KB
TestCase 1	Easy	Sample case	✔	Success	1	0.1295 sec	29.7 KB
TestCase 2	Easy	Sample case	✔	Success	1	0.2695 sec	30.1 KB
TestCase 3	Easy	Sample case	✔	Success	2	0.1606 sec	30.1 KB
TestCase 4	Easy	Hidden case	✔	Success	2	0.1412 sec	30.1 KB
TestCase 5	Easy	Sample case	✔	Success	2	0.2297 sec	30.1 KB
TestCase 6	Medium	Hidden case	✔	Success	4	0.2611 sec	30.5 KB
TestCase 7	Medium	Hidden case	✔	Success	4	0.1514 sec	30.6 KB
TestCase 8	Medium	Hidden case	✔	Success	4	0.3401 sec	34.6 KB
TestCase 9	Hard	Hidden case	✔	Success	7	0.205 sec	34.6 KB
TestCase 10	Hard	Hidden case	✔	Success	7	0.1848 sec	34 KB
TestCase 11	Hard	Hidden case	✔	Success	7	0.3962 sec	46.1 KB
TestCase 12	Hard	Hidden case	✔	Success	8	1.5595 sec	188 KB

No Comments

QUESTION 2



Correct Answer

Score 75

Team Formation 2 > Coding Medium Algorithms Problem Solving Combinatorics

Interviewer Guidelines

QUESTION DESCRIPTION

FC Codelona is trying to assemble a team from a roster of available players. They have a minimum number of players they want to sign, and each player needs to have a skill rating within a certain range. Given a list of players' skill levels with desired upper and lower bounds, determine how many teams can be created from the list.

Example

skills = [12, 4, 6, 13, 5, 10]

minPlayers = 3

minLevel = 4

maxLevel = 10

- The list includes players with skill levels [12, 4, 6, 13, 5, 10].
- They want to hire at least 3 players with skill levels between 4 and 10, inclusive.
- Four of the players with the following skill levels {4, 6, 5, 10} meet the criteria.
- There are 5 ways to form a team of at least 3 players : {4, 5, 6}, {4, 6, 10}, {4, 5, 10}, {5, 6, 10}, and {4, 5, 6, 10}.
- Return 5.

Function Description

Complete the function *countTeams* in the editor below.

countTeams has the following parameter(s):

int skills[n]: an array of integers that represent the skill level per player

int minPlayers: the minimum number of team members required

int minLevel: the lower limit for skill level, inclusive

int-maxLevel: the upper limit for skill level, inclusive

Return

int: the total number of teams that can be formed per the criteria

Constraints

- $1 \leq n \leq 20$
- $1 \leq \text{minPlayers} \leq n$
- $1 \leq \text{minLevel} \leq \text{maxLevel} \leq 1000$
- $1 \leq \text{skills}[i] \leq 1000$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n , the size of the array *skills*.

The next n lines each contain an element *skills[i]* where $0 \leq i < n$.

The next line contains an integer, *minPlayers*, the minimum number of players to be included in the team.

The next line contains an integer, *minLevel*, the lower limit of skill level to select

The next line contains an integer, *maxLevel*, the upper limit of skill level to select

▼ Sample Case 0

Sample Input 0

STDIN		Function
-----		-----
4	→	skills[] size n = 4
4	→	skills = [4, 8, 5, 6]
8		
5		
6		
1	→	minPlayers = 1
5	→	minLevel = 5
7	→	maxLevel = 7

Sample Output 0

3

Explanation 0

- The list includes players with skill levels [4, 8, 5, 6].
- They want to hire at least 1 player with skill levels between 5 and 7, inclusive.
- Two of the players *with the following skill levels* {5, 6} *meet the criteria*
- There are 3 ways to form a team of at least 1 player : {5}, {6}, {5, 6}.
- Returns 3.

▼ Sample Case 1

Sample Input 1

STDIN		Function
-----		-----
4	→	skills[] size n = 4
4	→	skills = [4, 8, 5, 6]
8		
5		
6		
2	→	minPlayers = 2
5	→	minLevel = 5
7	→	maxLevel = 7

Sample Output 1

1

Explanation 1

- The list includes players with skill levels $[4, 8, 5, 6]$.
- They want to hire at least 2 players with skill levels between 5 and 7, inclusive.
- Two of the players *with the following skill levels* $\{5, 6\}$ *meet the criteria*
- There is only one ways to form a team of at least 2 players : $\{5, 6\}$.
- Returns 1

▼ Sample Case 2

Sample Input 2

STDIN	Function
-----	-----
4	→ skills[] size n = 4
4	→ skills = [4, 8, 5, 6]
8	
5	
6	
2	→ minPlayers = 2
7	→ minLevel = 7
8	→ maxLevel = 8

Sample Output 2

0

Explanation 2

- The list includes players with skill levels $[4, 8, 5, 6]$.
- They want to hire at least 2 players with skill levels between 7 and 8, inclusive.
- One of the players *with the following skill levels* $\{8\}$ *meet the criteria*.
- There is no way to form a team of at least 2 players.
- Returns 0.

INTERVIEWER GUIDELINES

▼ Hint 1

Count the number of eligible players. Now you just need to count the number of ways you can choose at least minPlayers from the eligible players. Look over the constraint of n.

▼ Hint 2

We can say that the number of ways to choose at least minPlayers from the eligible players is equal to number of ways to choose exactly p players such that $p \geq \text{minPlayers}$.

▼ Solution

Concepts covered: Combinatorics

Optimal Solution:

Count the number of players such that their skill lies in the range $[\text{minLevel}, \text{maxLevel}]$. Let's suppose that this number is m, now you need to find the number of ways to choose at least minPlayers from these m players. We can break the problem into sum of the number of ways to choose exactly p

players from m players such that $p > \text{minPlayers}$. Now, the number of ways to choose p players from m players is equal to $C(m, p)$ or $m!/(p!(m-p)!)$.

```
def nCr(n, r):
    ans = 1
    for i in range(1, n+1):
        ans *= i
    for i in range(1, r+1):
        ans //= i
    for i in range(1, n-r+1):
        ans //= i
    return ans

def countTeams(skills, minPlayers, minLevel, maxLevel):
    count = sum(int(j >= minLevel and j <= maxLevel) for j in skills)
    ans = 0
    for i in range(minPlayers, count+1):
        ans += nCr(count, i)
    return ans
```

Brute Force Approach: Passes all test cases

```
def countTeams(skills, minPlayers, minLevel, maxLevel):
    if len(skills) == 0:
        return (minPlayers <= 0)
    ans = countTeams(skills[1:], minPlayers, minLevel, maxLevel)
    if skills[0] >= minLevel and skills[0] <= maxLevel:
        ans += countTeams(skills[1:], minPlayers - 1, minLevel, maxLevel)
    return ans
```

In the Brute force approach, we have built a recursive solution which tries every possible subset of the players. Since n is quite small, the brute force approach will work here as well.

Error Handling: Beware of overflows if you are using C++/Java/similar programming language which have specific size of integers. If you need to compute the value of $C(20, 10)$ you probably will have an overflow.

▼ Complexity Analysis

Time Complexity - $O(n^2)$.

Since the number of eligible players could be n in the worst case, and if the number of minPlayers is 1, you have to make $O(n)$ passes and each pass will take $O(n)$ time for computation.

Space Complexity - $O(1)$ - No extra space is required.

▼ Follow up Question

Solve the problem for n of the order of 10^5 , output the answer modulo M (prime number).

You could compute the value of $C(n, r)$ in $O(\log M)$ time by pre-computation of all the factorials and using Fermat's Theorem.

CANDIDATE ANSWER

Language used: **Java 8**






```
1 class Result {
2
3     /*
4      * Complete the 'countTeams' function below.
5      */
```



```

6      * The function is expected to return an INTEGER.
7
8      * The function accepts following parameters:
9      * 1. INTEGER_ARRAY skills
10     * 2. INTEGER minPlayers
11     * 3. INTEGER minLevel
12     * 4. INTEGER maxLevel
13     */
14
15     public static int countTeams(List<Integer> skills, int minPlayers, int
minLevel, int maxLevel) {
16         if(skills.size()!=0){
17             List<Integer> filter44=skills.stream().filter(i->(i>=minLevel &&
18 i<=maxLevel)).collect(Collectors.toList());
19             if(filter44.size()>minPlayers){
20                 int count44=1;
21                 for (int j = minPlayers; j < filter44.size(); j++) {
22                     BigInteger n44=factorial(j);
23                     BigInteger n_r=factorial(filter44.size()-j);
24                     BigInteger value=
25 factorial(filter44.size()).divide((n44.multiply(n_r)));
26                     count44+=value.intValue();
27                 }
28                 return count44;
29             }else{
30                 return (filter44.size()==minPlayers?1:0);
31             }
32         }
33         else{
34             return 0;
35         }
36
37
38
39     }
40     // static int factorial(int n){
41     //     return (n==1 || n==0)?1:n*factorial(n-1);
42     // }
43     public static BigInteger factorial(int n){
44         BigInteger factorial44=BigInteger.ONE;
45         for (int i = n; i > 0; i--) {
46             factorial44=factorial44.multiply(BigInteger.valueOf(i));
47         }
48         return factorial44;
49     }
50
51 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0: $O(2^n)$ or $O(n^2)$ or $O(n)$	Easy	Sample case	 Success	1	0.1868 sec	30.2 KB
Testcase 1: $O(2^n)$ or $O(n^2)$ or $O(n)$	Easy	Sample case	 Success	1	0.1803 sec	30.2 KB
Testcase 2: $O(2^n)$ or $O(n^2)$ or $O(n)$	Easy	Sample case	 Success	1	0.1392 sec	30.3 KB
Testcase 3: $O(2^n)$ or $O(n^2)$ or Testcase 3: $O(n)$	Medium	Sample case	 Success	10	0.1478 sec	29.8 KB
Testcase 4: $O(2^n)$ or $O(n^2)$ or $O(n)$	Medium	Hidden case	 Success	10	0.1351 sec	30.3 KB

Testcase 5: $O(2^n)$ or $O(n^2)$ or $O(n)$	Medium	Hidden case	Success	10	0.2554 sec	30.3 KB
Testcase 6: $O(2^n)$ or $O(n^2)$ or $O(n)$	Medium	Hidden case	Success	10	0.2234 sec	30.6 KB
Testcase 7: $O(2^n)$ or Testcase 7: $O(n^2)$ or Testcase 7: $O(n)$	Medium	Hidden case	Success	10	0.2623 sec	30.4 KB
Testcase 8: $O(2^n)$ or $O(n^2)$ or $O(n)$	Hard	Hidden case	Success	10	0.2193 sec	30.3 KB
Testcase 9: $O(2^n)$ or $O(n^2)$ or $O(n)$	Hard	Hidden case	Success	11	0.1429 sec	30.3 KB
Testcase 10: $O(2^n)$ or $O(n^2)$ or $O(n)$ Answer = 0 (No teams satisfy the criteria)	Easy	Sample case	Success	1	0.1546 sec	30.1 KB

No Comments

QUESTION 3



Needs Review

Score 50

Distance Between Two Points > Coding

C++

Java

C#

Easy

OOPS

Implementation

QUESTION DESCRIPTION

This challenge involves points in two and three dimensional space. The classes and methods to implement will store values for coordinates as well as calculate distances between points. The *2D* and *3D* distances between two points are calculated using the following formulae:

$$\text{2D distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\text{3D distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Implement the classes and methods defined below:

1. A superclass named *Point2D*:

Instance Variables	
Name	Functionality
x	Stores the value of the x-coordinate.
y	Stores the value of the y-coordinate.

Constructor	
Name	Functionality
<i>Point2D(...)</i>	A parameterized constructor that initializes the instance variables.

Methods	
Name	Functionality
	Calculates and returns the <i>2D</i> distance

<i>double dist2D(Point2D p)</i>	between two points (the current <i>Point2D</i> object and <i>Point2D</i> parameter <i>p</i>).
<i>void printDistance(double d)</i>	Prints the <i>2D</i> distance between two points as <i>2D distance = k</i> , where <i>k</i> is distance <i>d</i> as a ceiling-rounded integer, on a new line.

2. A derived class named *Point3D* that extends *Point2D*:

Instance Variable	
Name	Functionality
<i>z</i>	Stores the value of the <i>z</i> -coordinate.

Constructor	
Name	Functionality
<i>Point3D(...)</i>	A parameterized constructor that initializes the instance variables.

Methods	
Name	Functionality
<i>double dist3D(Point3D p)</i>	Calculates and returns the <i>3D</i> distance between two points (the current <i>Point3D</i> object and <i>Point3D</i> parameter <i>p</i>).
<i>void printDistance(double d)</i>	Prints the <i>3D</i> distance between two points as <i>3D distance = k</i> , where <i>k</i> is distance <i>d</i> as a ceiling-rounded integer, on a new line.

A *main* method is provided in the locked portion of the editor. It parses six values representing point coordinates and calls the implemented constructors and methods. Here, *x[1]*, *y[1]*, and *z[1]* represent the coordinates of the *first* point, and *x[2]*, *y[2]*, and *z[2]* represent the coordinates of the *second* point. Note that printed output must exactly match the above for the test cases to pass.

Constraints

- $-128 \leq x, y, z \leq 127$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

Line: description

- x[1]*: integer
- y[1]*: integer
- z[1]*: integer
- x[2]*: integer
- y[2]*: integer
- z[2]*: integer

▼ Sample Case 0

Sample Input 0

```
STDIN  Function
-----  -
1      → x[1] = 1
2      → y[1] = 2
3      → z[1] = 3
4      → x[2] = 4
5      → y[2] = 5
6      → z[2] = 6
```

Sample Output 0

```
2D distance = 5
3D distance = 6
```

Explanation 0

For the first point: $x = 1, y = 2, z = 3$.

For the second point: $x = 4, y = 5, z = 6$.

The formula gives a 2D distance of $\sqrt{3^2 + 3^2} = 4.242640687119285$. The ceiling is 5.

The formula gives a 3D distance of $\sqrt{3^2 + 3^2 + 3^2} = 5.196152422706632$. The ceiling is 6.

▼ Sample Case 1

Sample Input 1

```
STDIN  Function
-----  -
-1     → x[1] = -1
-2     → y[1] = -2
4      → z[1] = 4
-7     → x[2] = -7
-1     → y[2] = -1
-1     → z[2] = -1
```

Sample Output 1

```
2D distance = 7
3D distance = 8
```

Explanation 1

For the first point, $x = -1, y = -2, z = 4$

For the second point, $x = -7, y = -1, z = -1$

The differences are $x: -6, y: 1, z: -5$

The squares of differences are $x: 36, y: 1, z: 25$

2D distance calculation: $\sqrt{36+1} = 6.08276253$, ceiling = 7

3D distance calculation: $\sqrt{36+1+25} = 7.874007874$, ceiling = 8

CANDIDATE ANSWER

Language used: **Java 8**

```
1  /* Write your class implementations here. Do not use access modifiers when
2  declaring your classes. */
3  class Point2D{
4      double x_44;
5      double y_44;
6      public Point2D(double x_44,double y_44){
7          this.x_44=x_44;
8          this.y_44=y_44;
```

```

9      }
10     public double dist2D(Point2D p){
11         double d1=(p.x_44-x_44)*(p.x_44-x_44);
12         double d2=(p.y_44-y_44)*(p.y_44-y_44);
13         double d3=d1+d2;
14         return Math.sqrt(d3);
15     }
16     public void printDistance(double d){
17         System.out.println("2D distance = "+(int)Math.ceil(d));
18     }
19
20
21 }
22 class Point3D extends Point2D{
23     double z_44;
24     public Point3D(double x_44,double y_44,double z_44){
25         super(x_44, y_44);
26         this.z_44=z_44;
27     }
28     public double dist3D(Point3D p){
29         double d1=(p.x_44-x_44)*(p.x_44-x_44);
30         double d2=(p.y_44-y_44)*(p.y_44-y_44);
31         double d3=(p.z_44-z_44)*(p.z_44-z_44);
32         double d4=d1+d2+d3;
33         return Math.sqrt(d4);
34     }
35     public void printDistance(double d){
36         System.out.println("3D distance = "+(int)Math.ceil(d));
37     }
38
39 }

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	2	0.1175 sec	25.1 KB
TestCase 1	Easy	Sample case	✔ Success	1	0.1117 sec	25.1 KB
TestCase 2	Easy	Sample case	✔ Success	6	0.1263 sec	24.9 KB
TestCase 3	Easy	Sample case	✔ Success	6	0.1882 sec	25.2 KB
TestCase 4	Easy	Hidden case	✔ Success	6	0.178 sec	25.1 KB
TestCase 5	Easy	Hidden case	✔ Success	6	0.1038 sec	25.1 KB
TestCase 6	Easy	Hidden case	✔ Success	6	0.1085 sec	25.3 KB
TestCase 7	Easy	Hidden case	✔ Success	8	0.1243 sec	25.1 KB
TestCase 8	Easy	Hidden case	✔ Success	9	0.0911 sec	25.1 KB

No Comments

QUESTION 4



Correct Answer

Score 50

Scheduling Errors > DbRank

SQL

Easy

Interviewer Guidelines

Simple Joins

Databases

QUESTION DESCRIPTION

A university maintains data on professors, departments, courses, and schedules in four tables: *DEPARTMENT*, *PROFESSOR*, *COURSE*, and *SCHEDULE*.

Write a query to print the names of professors with the names of the courses they teach (or have taught)

outside of their department. Each row in the results must be distinct (i.e., a professor teaching the same course over multiple semesters should only appear once), but the results can be in any order. Output should contain two columns: *PROFESSOR.NAME*, *COURSE.NAME* .

PROFESSOR			DEPARTMENT	
Name	Type	Description	Description	
ID	Integer	A professor's ID in the inclusive range [1, 1000]. This is a <i>primary key</i> .	A department's ID in the inclusive range [1, 1000]. This is a <i>primary key</i> .	
NAME	String	A professor's name. This field contains between 1 and 100 characters (inclusive).	A department's name. This field contains between 1 and 100 characters (inclusive).	
DEPARTMENT_ID	Integer	A professor's department ID. This is a <i>foreign key</i> to <i>DEPARTMENT.ID</i> .	A department's salary in the inclusive range [5000, 40000].	
SALARY	Integer	A professor's salary in the inclusive range [5000, 40000].	A department's salary in the inclusive range [5000, 40000].	

l
m
a
r
y
k
e
y
.

A
d
e
p
a
r
t
m
e
n
t
n
a
m
e
.

T
h
i
s
f
i
e
l
d
c
o
n
t
a
i
n
s
b

nothing

t
w
e
e
n
1
a
n
d
1
0
0
c
h
a
r

a
c
t
e
r
s
(
i
n
c
l
u
s
i
v
e
)
.

COURSE			SCHEDULE		
Name	Type	Description	Name	Type	Description
ID	Integer	A course ID in the inclusive range [1, 1000]. This is a <i>primary key</i> .	PROFESSOR_ID	Integer	The ID of the professor teaching the course. This is a <i>foreign key</i> to <i>PROFESSOR.ID</i> .
NAME	String	A course name. This field contains between 1 and 100 characters (inclusive).	COURSE_ID	Integer	The course's ID number. This is a <i>foreign key</i> to <i>COURSE.ID</i> .
DEPARTMENT_ID	Integer	A course's department ID. This is a <i>foreign key</i> to <i>DEPARTMENT.ID</i> .	SEMESTER	Integer	A semester ID in the inclusive range [1, 6].
CREDITS	Integer	The number of credits allocated to the course in the inclusive range [1, 10].	YEAR	Integer	A calendar year in the inclusive range [2000, 2017].

Output Format

Each distinct row of results must contain the name of a professor followed by the name of a course they taught outside of their department in the format:

```
PROFESSOR.NAME COURSE.NAME
```


▼ Sample Data Tables

PROFESSOR	DEPARTMENT
DEPARTMENT_ID	NAME
14169 Daniels	Biological Sciences
34793 Knight	Technology
345194 Myers	Humanities & Social Sciences
49686 Antonio Rodriguez	Clinical Medicine
5210810 Gome	Arts and Humanities
610487 George	Physical Sciences
74351a Vasquez	
825726 Flores	
925678 Gilbert	
126642 New Stevens	

COURSE	SCHEDULE
DEPARTMENT_ID	SECTION_ID
Clinical Biochemistry	42003
Astronomy	32011
Clinical Neuroscience	12011
Pure Mathematics and Mathematical Statistics	73010s
Geography	42001
Chemistry	32012
Physics	12009
Earth Science	12014
Materials Science and Metallurgy	12008
Applied Mathematics and Theoretical Physics	12003

Sample Output

Antonio Rodriguez Astronomy
Harry Myers Earth Sciences
Nancy Daniels Materials Science and Metallurgy
Gloria Vasquez Materials Science and Metallurgy
Antonio Rodriguez Geography
Daniel Gilbert Earth Sciences
Matthew Stevens Applied Mathematics and Theoretical Physics
Nancy Daniels Pure Mathematics and Mathematical Statistics
Nancy Daniels Applied Mathematics and Theoretical Physics
Nancy Daniels Materials Science and Metallurgy

Explanation

By referring to the sample data above, we can confirm that:

1. Professor *Antonio Rodriguez's department_id* is 3, but the *Astronomy* course's *department_id* is 1.
2. Professor *Harry Myers's department_id* is 4, but the *Earth Sciences* course's *department_id* is 1.
3. Professor *Nancy Daniels's department_id* is 4, but the *Astronomy* course's *department_id* is 1.
4. Professor *Gloria Vasquez's department_id* is 4, but the *Materials Science and Metallurgy* course's *department_id* is 1.
5. Professor *Antonio Rodriguez's department_id* is 3, but the *Geography* course's *department_id* is 1.
6. Professor *Daniel Gilbert's department_id* is 5, but the *Earth Sciences* course's *department_id* is 1.
7. Professor *Matthew Stevens's department_id* is 2, but the *Applied Mathematics and Theoretical Physics* course's *department_id* is 1.
8. Professor *Nancy Daniels's department_id* is 4, but the *Pure Mathematics and Mathematical Statistics* course's *department_id* is 1.
9. Professor *Nancy Daniels's department_id* is 4, but the *Applied Mathematics and Theoretical Physics* course's *department_id* is 1.
10. Professor *Nancy Daniels's department_id* is 4, but the *Materials Science and Metallurgy* course's *department_id* is 1.

INTERVIEWER GUIDELINES**▼ Solution**

Concepts covered: JOIN, DISTINCT

Solution:

Join 3 tables - professor, schedule, and course using schedule for the many to many relation. The WHERE clause limits result to departments other than the professor's. Finally, apply the DISTINCT directive to remove duplicates.

```
SELECT DISTINCT p.name AS professor_name, c.name AS course_name
FROM professor p
JOIN schedule s ON s.professor_id = p.id
JOIN course c ON s.course_id = c.id
WHERE c.department_id <> p.department_id;
```

Omitting the keyword DISTINCT leads to incorrect result.

CANDIDATE ANSWER

Language used: **MySQL**

```
1  /*
2  Enter your query here.
3  Please append a semicolon ";" at the end of the query
4  */SELECT distinct p44.name,c44.name FROM PROFESSOR as p44,COURSE as c44
5  ,SCHEDULE as s44 WHERE s44.professor_id=p44.id AND s44.course_id=c44.id AND
   p44.department_id<>c44.department_id;
```

Time taken: **0.15 sec**

No Comments

QUESTION 5



Needs Review

Score 12

Sprint Training

> Coding

Data Structures

Medium

Algorithms

Arrays

Problem Solving

Dynamic Programming

Interviewer Guidelines

QUESTION DESCRIPTION

Pat is an ordinary kid who works hard to be a great runner. As part of training, Pat must run sprints of different intervals on a straight trail. The trail has numbered markers that the coach uses as goals. Pat's coach provides a list of goals to reach in order. Each time Pat starts at, stops at, or passes a marker it is considered a *visit*. Determine the lowest numbered marker that is visited the most times during Pat's day of training.

Example

 $n = 5$ $sprints = [2, 4, 1, 3]$

if the number of markers on the trail, $n = 5$, and assigned $sprints = [2, 4, 1, 3]$, Pat first sprints from position $2 \rightarrow 4$. The next sprint is from position $4 \rightarrow 1$, and then $1 \rightarrow 3$. A marker numbered position p is considered to be *visited* each time Pat either starts or ends a sprint there and each time it is passed while sprinting. The total number of visits to each position in the example is calculated like so:

Total Visits Per Position					
Sprint	1	2	3	4	5
$2 \rightarrow 4$		☺ \rightarrow	\rightarrow	\rightarrow ☺	
$4 \rightarrow 1$	☺ \leftarrow	\leftarrow	\leftarrow	\leftarrow ☺	
$1 \rightarrow 3$	☺ \rightarrow	\rightarrow	\rightarrow ☺		
Total Visits	2	3	3	2	0

Pat has visited markers 2 and 3 a total of 3 times each. Since $2 < 3$, the lowest numbered marker that is visited the most times during Pat's day of training is 2.

Function Description

Complete the function *getMostVisited* in the editor below.

getMostVisited has the following parameter(s):

int n: an integer denoting the number of markers along the trail

int sprints[m]: an array of integers denoting the sequence of markers to reach, beginning at the marker shown in *sprints[0]*.

Returns

int: an integer denoting Pat's *most visited* position on the trail after performing all $m - 1$ sprints. If there are multiple such answers, return the smallest one.

Constraints

- $1 \leq n \leq 10^5$
- $2 \leq m \leq 10^5$
- $1 \leq sprints[i] \leq m$ (where $0 \leq i < m$)
- $sprints[i-1] \neq sprints[i]$ (where $0 < i < m$)

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n , the number of markers along the path.
The second line contains an integer m , the number of markers in the list of goals.
The next m lines each contain an element $sprints[i]$ where $0 \leq i < m$.

▼ Sample Case 0

Sample Input 0

```
STDIN      Function Parameters
-----      -
10      →  n = 10
4      →  sprints[] size m = 4
1      →  sprints = [1, 5, 10, 3]
5
10
3
```

Sample Output 0

```
5
```

Explanation 0

Given $sprints = [1, 5, 10, 3]$, Pat performs the following sequence of sprints:

Sprint	1	2	3	4	5	6	7	8	9	10
1 → 5	☹ →	→	→	→	→ ☹					
5 → 10					☹ →	→	→	→	→	→ ☹
10 → 3			☹ ←	←	←	←	←	←	←	← ☹
Total Visits	1	1	2	2	3	2	2	2	2	2

In the table above, Pat visited marker 5 the most.

▼ Sample Case 1

Sample Input 1

```
STDIN      Function Parameters
-----      -
5      →  n = 5
2      →  sprints[] size m = 2
1      →  sprints = [1, 5]
5
```

Sample Output 1

```
1
```

Explanation 1

Given $sprints = [1, 5]$, Pat performs the following sprint:

Sprint	1	2	3	4	5
1 → 5	☹ →	→	→	→	→ ☹
Total Visits	1	1	1	1	1

In the table above, every marker is visited the same number of times. Return the smallest of these

In the table above, every marker is visited the same number of times. Return the smallest of these, which is 1.

▼ Sample Case 2

Sample Input 2

```
STDIN      Function Parameters
-----
9          → n = 9
4          → sprints[] size m = 4
9          → sprints = [9, 7, 3, 1]
7
3
1
```

Sample Output 2

3

Explanation 2

Given *sprints* = [9, 7, 3, 1], Pat performs the following sequence of sprints:

Sprint	1	2	3	4	5	6	7	8	9
9 → 7							☺ ←	←	← ☺
7 → 3			☺ ←	←	←	←	← ☺		
3 → 1	☺ ←	←	← ☺						
Total Visits	1	1	2	1	1	1	2	1	1

In the table above, Pat visited positions 3 and 7 the most. Return the smallest of these, which is 3.

INTERVIEWER GUIDELINES

▼ Solution

Skills: Arrays, Algorithms, Problem Solving

Optimal Solution:

The problem can be solved by using a difference array *D* of size $(n+1)$. Initially, all the elements in the array are equal to 0. For each sprint between *l* and *r* ($l < r$), we add 1 to *D*[*l*] and subtract it from *D*[*r*+1], i.e., we do *D*[*l*] += 1, *D*[*r*+1] -= 1. Once all the sprints are done, we do *D*[*i*] = *D*[*i*-1] + *D*[*i*] for all *i* (*i*=1...*n*) to obtain the array representing the number of visits to each marker *i*. We then iterate over this array to obtain the smallest numbered marker with the maximum number of visits.

```
def getMostVisited(n, sprints):
    # Write your code here

    #Creating difference array
    diff = [0 for i in range(n+2)]
    m = len(sprints)
    for i in range(1,m):
        st,en = min(sprints[i-1],sprints[i]),max(sprints[i-1],sprints[i])
        diff[st] += 1
        diff[en+1] -= 1

    #Preparing array representing the number of times a marker is visited
    for i in range(1,n+1):
        diff[i] += diff[i-1]
    mxm = max(diff[1:n+1])
```

```
for i in range(1,n+1):
    if(diff[i] == mxm):
        return i
return 0
```

▼ Complexity Analysis

Time Complexity - $O(n)$

Space Complexity - $O(n)$

CANDIDATE ANSWER

Language used: **Java 8**

```
1  class Result {
2
3      /*
4       * Complete the 'getMostVisited' function below.
5       *
6       * The function is expected to return an INTEGER.
7       * The function accepts following parameters:
8       * 1. INTEGER n
9       * 2. INTEGER_ARRAY sprints
10      */
11
12     public static int getMostVisited(int n, List<Integer> sprints) {
13         Map<Integer,Integer> map44=new HashMap<Integer,Integer>();
14         for (int i = 1; i < n+1; i++) {
15             map44.putIfAbsent(i, 0);
16
17         }
18         for (int i = 0; i < sprints.size()-1; i++) {
19             int a44=sprints.get(i);
20             int b44=sprints.get(i+1);
21             if(a44<b44){
22                 for (int j = a44; j <=b44; j++) {
23                     map44.put(j, map44.get(j)+1);
24                 }
25             }
26             else{
27                 for (int j = b44; j <=a44; j++) {
28                     map44.put(j, map44.get(j)+1);
29                 }
30             }
31         }
32         int output=0;
33         int maxVisits=0;
34         for (Map.Entry<Integer,Integer> entry:map44.entrySet()) {
35             int key=entry.getKey();
36             int value=entry.getValue();
37             if(maxVisits<value){
38                 output=key;
39                 maxVisits=value;
40             }else if(maxVisits==value && output>key){
41                 output=key;
42             }
43         }
44         return output;
45     }
```

46
47
48
49
50

}

}

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	1	0.1472 sec	30.2 KB
TestCase 1	Easy	Sample case	✔ Success	1	0.1404 sec	30 KB
TestCase 2	Easy	Sample case	✔ Success	1	0.1522 sec	30 KB
TestCase 3	Easy	Sample case	✔ Success	3	0.4359 sec	41.6 KB
TestCase 4	Easy	Hidden case	✔ Success	3	0.5697 sec	50.2 KB
TestCase 5	Easy	Sample case	✔ Success	3	1.676 sec	200 KB
TestCase 6	Medium	Hidden case	✘ Terminated due to timeout	0	4.145 sec	214 KB
TestCase 7	Medium	Hidden case	✘ Terminated due to timeout	0	4.0372 sec	212 KB
TestCase 8	Medium	Hidden case	✘ Terminated due to timeout	0	4.0116 sec	216 KB
TestCase 9	Hard	Hidden case	✘ Terminated due to timeout	0	4.1662 sec	213 KB
TestCase 10	Hard	Hidden case	✘ Terminated due to timeout	0	4.0478 sec	213 KB
TestCase 11	Hard	Hidden case	✘ Terminated due to timeout	0	4.0565 sec	212 KB
TestCase 12	Hard	Hidden case	✘ Terminated due to timeout	0	4.0157 sec	220 KB

No Comments