BUBT Intra-University Programming Contest, Fall 2019-20 (Senior Division)

Organized By:

Dept. of CSE, BUBT

Contest Rules

* Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the results**.** ***Submitted codes should not contain team name and the file name should not have any white space.***

* Notification of accepted or rejected runs will be suspended at the last hour of the contest time to keep the final results secret.

* A contestant may submit a clarification request to judges only through the Codeto.Win clarification system. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants. Judges may prefer not to answer a clarification at all in which case that particular clarification request will be marked as IGNORED in the Codeto.Win clarification page.

* Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee while seated at the team desk. **They cannot even talk with their team members when they are walking around the contest floor to have food or any other purpose.** Systems support staff or judges may advise contestants on system-related problems such as explaining system error messages.

* While the contest is scheduled for a particular time length (four/five hours), the contest director has the authority to alter the length of the contest in the event of unforeseen difficulties. If the contest duration is altered, every attempt will be made to notify contestants in a timely and uniform manner.

* **A team may be disqualified by the Contest Director** for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior, browsing internet except **Codeto.Win** site or communicating with other teams. **The judges on the contest floor** will report to the **Judging Director** about distracting behavior of any team. **The judges can also recommend penalizing a team with additional penalty minutes for their distracting behavior.**

* Ten, Eleven or Twelve problems will be posed. So far as possible, problems will avoid dependence on detailed knowledge of a particular application area or particular contest language.

* Contestants will have foods available in their contest room during the contest. So they cannot leave the contest room during the contest without explicit permission from the judges. **The contestants are not allowed to communicate with any contestant (even contestants of his own team) when they are outside the contest arena.**

* Teams can bring up to 1**00 pages of printed materials** with them. They can also bring five additional books, but they are not allowed to bring calculators or any machine-readable devices like CD, DVD, Pen-drive, iPod, MP3/MP4 players, floppy disks, etc. **Mobile phones MUST be switched off at all times and stored inside a bag or any other place that is publicly non-visible during the entire contest time. Failure to adherence to this clause under any condition will very likely lead to strict disciplinary retaliation and possible disqualification.**

*** The decision of the judges is final.**

*** Teams should inform the volunteers/judges if they don't get a reply from the judges within 10 minutes of submission. Teams should also notify the volunteers if they cannot log in to the Codeto.Win system. These sorts of complain will not be entertained after the contest.**

Problem Setters

01. ***Md. Saifur Rahman**, Assistant Professor, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

02. ***Md. Murad Al Wajed,** Intake 26, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)
    Software Engineer, Tapstar Apps

03. ***Abu Quwsar Ohi,** Intake 33, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

04. ***Nazmul Hoda,** Intake 33, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

05. ***Nazmul Alam Nayan,** Intake 35, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

06. ***Md. Elias Hassan Naim,** Intake 32, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

07. ***Maruf Ahmed Rahad,** Intake 35, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

08. ***Tanbir Ahmed,** Intake 35, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

09. ***M M Mehedi Hasan,** Intake 33, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

10. ***Walid Khan Jim,** Intake 35, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

11. ***Md. Farid Miah,** Intake 32, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

12. ***Md. Mimsad Ahmed Hridoy,** Intake 33, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

13. ***Monzurul Islam,** Intake 35, Dept. of CSE*
    Bangladesh University of Business and Technology (BUBT)

Judge Panel Members

1. ***Md. Saifur Rahman,*** *Assistant Professor, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

2. ***Md. Ashraful Islam****, Lecturer, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

3. ***Md. Elias Hassan Naim,*** *Intake 32, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

4. ***Md. Farid Miah,*** *Intake 32, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

5. ***Nazmul Alam Nayan,*** *Intake 35, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

6. ***Abu Quwsar Ohi,*** *Intake 33, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

7. ***M M Mehedi Hasan,*** *Intake 33, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

8. ***Maruf Ahmed Rahad,*** *Intake 35, Dept. of CSE*
   Bangladesh University of Business and Technology (BUBT)

A. Welcome to BUBT IUPC

Input: Standard Input, **Output:** Standard Output

Time Limit: 1 second(s)

Memory Limit: 256 megabytes

Problem Statement:

There is no statement for this problem. Just copy the code and submit it.

```
#include <stdio.h>
int main() {
	printf("Welcome to BUBT IUPC\n");
	return 0;
}
```

Input:

There is no input.

Output:

Print “Welcome to BUBT IUPC” in a single line.

Sample Input/Output:

Sample Input	Sample Output
	Welcome to BUBT IUPC

B. Three Consecutive Numbers

Input: Standard Input, **Output:** Standard Output
Time Limit: 1 second(s)
Memory Limit: 256 megabytes

Problem Statement:

Given a number **N**. You have to find out three integers **X**, **Y**, and **Z** where,

1. **X = Y – 1**,
2. **Z = Y + 1**, and
3. **X + Y + Z = N**.

Input:

Every input contains a positive number **N** ($6 \leq \mathbf{N} \leq 1000$).

Output:

If no valid solution exists, print **-1**. Otherwise, print **X**, **Y**, and **Z** in a line and print a space between every two consecutive integers. Check sample input/output for clarity.

Sample Input/Output:

Sample Input	Sample Output
6	1 2 3

Problem Setter: Md. Elias Hassan Naim, Intake 32, Dept. of CSE.

C. Hardest Calculation Ever!!

Input: Standard Input, **Output:** Standard Output
Time Limit: 1 second(s)
Memory Limit: 256 megabytes

Problem Statement:

GCD stands for Greatest Common Divisor and LCM stands for Lowest Common Multiple. You are given an equation **GCD(X, Y) + LCM(X, Y) = X+Y** where **X** and **Y** are two integers. Now your task is to check the equation is valid or not for the given **X** and **Y**.

Input:

The first line contains one integer **T** $(1 \leq T \leq 10^6)$, denoting the number of test cases.

Then **T** lines follow, each describing a test case. Each line contains two integers **X** and **Y** $(0 < X,\ Y \leq 10^{18})$.

Output:

For each test case, if the equation is satisfied, print "Valid" and then print the value of **GCD(X, Y) + LCM(X, Y)** in the next line. If the equation is not satisfied print “Invalid”.

Sample Input/Output:

Sample Input	Sample Output
3	Invalid
5 7	Valid
10 20	30
2 3	Invalid

D. Little Hackers

Input: Standard Input, **Output:** Standard Output
Time Limit: 1 second(s)
Memory Limit: 256 megabytes

Problem Statement:

Little hackers Andrew and Beth are trying to find some secret information in a long log of the chat conversation of their friend Chris. They know that the secret part of the conversation is related to the message to his friend Dan they intercepted before. But the log is too long to search by hand. Therefore they decided to find the parts that are most similar to the message to analyze them later. They ask you to help them to write the corresponding program.

Let us represent the log as one string **t** with all spaces, line feeds, digits and punctuation marks deleted. We will also ignore the case of the letters. Thus, the string only consists of capital letters of the English alphabet. The message, in turn, is also converted to the same form, let us denote the corresponding string as **p**.

For each position, **i** in **t** Andrew and Beth want to find the length of the maximal substring of **p** that starts in **t** at **i'th** position. Thus, for each **i** you have to find such **j = f(i)** that **t[i . . . i+j -1] = p[k . . . k +j -1]** for some **k**, and **j** is maximal possible.

Input:

The input file contains two lines. The first line contains **t**, the second line contains **p**. The length of **t** does not exceed **100**, the length of **p** does not exceed **100**. Both strings are not empty.

Input only consists of capital letters of the English alphabet.

Output:

Let the length of **t** be **n**. Output n numbers — for each **i** output the corresponding **f(i)**.

Sample Input/Output:

Sample 1:

Sample Input	Sample Output
ABBAABABC ABAAB	2 1 4 3 3 2 2 1 0

Sample 2:

Sample Input	Sample Output
AAAAAAAA AAAAAAAAAA	8 7 6 5 4 3 2 1

E. Selecting Guard (II)

Input: Standard Input, **Output:** Standard Output
Time Limit: 1 second(s)
Memory Limit: 256 megabytes

Problem Statement:

BUBT is a famous university. It's CSE department is very advanced because they produce good programmers nowadays. They arrange a programming contest in every semester. But this time the programmers don't want to participate in the contest by risking their lives. That's why Dipu sir set some students to guard the streets (which are outside of BUBT) for the programmers' safety. There are lots of stations and streets (bi-directional). The students have to guard all the stations and streets. A student can guard the streets and stations adjacent to it. But the students are not well behaved. If a street is guarded by multiple students they start fighting. So Dipu sir asked Zabir to write a program that finds the minimum number of students to guard all the stations and roads. Since Zabir is not a good programmer he needs your help.

Input:

The first line of the input contains a single integer **T** ($\boldsymbol{T} \leq 100$), indicating the number of test cases. Each test case begins with 2 integers **n** ($1 \leq \boldsymbol{n} \leq 250$) and **m** ($0 \leq \boldsymbol{m} \leq 10000$). Here, **n** is the number of stations and **m** is the number of streets. Each of the next **m** lines contains 2 integers **a** and **b** denoting that there is a street between the stations **a** and **b**. All the stations are numbered from **0** to **n−1**.

Output:

For each test case output in a single line an integer **X** denoting the minimum number of guards needed to guard all the stations and streets. If not possible print **-1**.

Sample Input/Output:

Sample Input	Sample Output

2
3 2
0 1
1 2
5 5
0 1
1 2
2 3
3 4
0 4
	1
	-1

Problem Setter: Maruf Ahmed Rahad, Intake 35, Dept. of CSE.

F. Rectangle Counting

Input: Standard Input, **Output:** Standard Output
Time Limit: 1 second(s)
Memory Limit: 256 megabytes

Problem Statement:

In the initial version or v1.0.0, you are given an **N*M** grid then count the number of rectangles in this grid by using the simple equation $\frac{(M)(M+1)(N)(N+1)}{4}$. In the next version v1.1.0, you are given **N** points then calculate how many rectangles are formed by these points. It is guaranteed that all rectangles are parallel to the axis.

Example: (0,0), (0,1), (1,1), (1,0), (2,0) and (2,1) number of rectangles are 3. Left rectangle (0,0), (0,1), (1,1), (1,0) right rectangle (1,0), (1,1), (2,1), (2,0) and big rectangle (0,0), (0,1), (2,1), (2,0).

Input:

The first line contains one integer N ($1 \leq N \leq 1000$) - the number of points. Then next N lines contain X and Y ($-10^9 \leq X, Y \leq 10^9$)- Coordinate of the X and Y-axis.

Output:

Calculate the number of rectangles is formed/shaped by these **N** points.

Sample Input/Output:

Sample 1:

Sample Input	Sample Output

6
0 0
0 1
1 1
1 0
2 0
2 1
	3

Sample 2:

Sample Input	Sample Output

4
0 0
0 1
1 0
1 2
	0

Problem Setter: Md. Farid Miah, Intake 32, Dept. of CSE.

G. Mr. MMMH and Suborno

Input: Standard Input, **Output:** Standard Output
Time Limit: 5 second(s)
Memory Limit: 256 megabytes

Problem Statement:

Mr. MMMH is a shopkeeper. He sells **n** types of chocolates, from **1** to **n** $(1 \leq \mathbf{n} \leq 10^7)$. She has **K** amount of money. She wants to buy the maximum number of **types** of chocolates.

The cost of chocolates are,

The cost of type 1 = The summation of divisors of 1 = 1.

The cost of type 2 = The summation of divisors of 2 = 1 + 2 = 3.

The cost of type 3 = The summation of divisors of 3 = 1 + 3 = 4.

And so on….

Check the Input/Output section for further information.

Input:

You are given two integers **n** $(1 \leq \boldsymbol{n} \leq 10^7)$ and **q** $(1 \leq \mathbf{q} \leq 10^5)$. Here, **n** is the number of types of chocolates, and **q** is the number of test cases.

In each case, you are given one integer **k** $(1 \leq \boldsymbol{k} \leq 10^{16})$, which is the amount of the money of Suborno.

Output:

You should find out, the maximum number of types of chocolates she can buy using **k** amount of money from **1** to **n** types of chocolates under the above-illustrated cost conditions.

Sample Input/Output:

Sample Input	Sample Output

5 5
1
3
8
1000000
7
	1
	1
	3
	5
	2

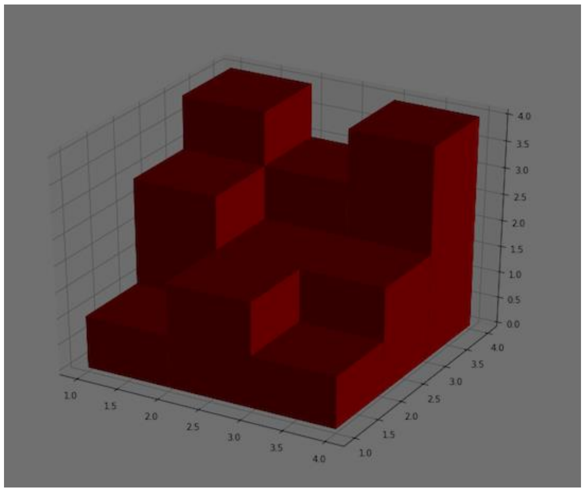
Problem Setter: M M Mehedi Hasan, Intake 33, Dept. of CSE.

H. Cost Count

Input: Standard Input, **Output:** Standard Output
Time Limit: 1 second(s)
Memory Limit: 256 megabytes

Problem Statement:

Mr. Samir likes to play with blocks. A block has 6 sides. Left, Right, Front, Back, Top, and Bottom. The size of a block is **(1*1*1)**. He makes building on a **(L*W)** square unit area. But in every coordinate, the number of blocks can be different.



Now he wants to recolor only one side of the building. You need to calculate the minimum cost and help him to choose the side of the building.

Input:

The first line contains the value of **W** (**W** <101) and (**L** <101).

The next **W** lines contain **L** space-separated integers.

The j^{th} integer in i^{th} line denotes the number of blocks (<101) in **(i,j)** position.

Output:

Just print the minimum cost to paint only one side of the building (open-sided block).

Sample Input/Output:

Sample 1:

Sample Input	Sample Output

3 3
1 2 1
3 2 2
4 3 4
	10

Sample 2:

Sample Input	Sample Output

3 3
1 1 1
1 1 1
1 1 1
	3

Problem Setter: Md. Murad Al Wajed, Intake 26, Dept. of CSE.

I. K'th Element (II)

Input: Standard Input, **Output:** Standard Output
Time Limit: 2 second(s)
Memory Limit: 256 megabytes

Problem Statement:

You are given **N** values and **Q** queries. Each query consists of three integers, **l**, **r**, and **k**. Each query states that you have to find the k'th value from the given input starting from index **l** to index **r**. Note that the index of **N** values starts from **1**.

Input:

The input starts with an integer **N** ($N \leq 10^5$), defining the number of values. The next line contains **N** integer values (0 <= values <= 10^9). The next line contains an integer **Q** ($Q \leq 10^5$), defining the number of queries.

The next **Q** lines contain three space-separated integer **l**, **r**, **k**, ($l \leq r$ and $k \leq r - l + 1$) defining that you have to find the k'th element from the input value that lies in index **l**, to **r**. By k'th value, it is meant that if you sort the values starting from index **l** to **r**, then you have to output the k'th value from the sorted values.

Output:

For each query, output the k'th value in a separated line.

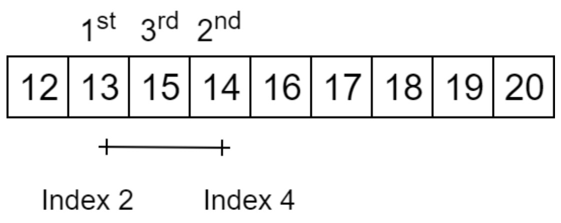
Sample Input/Output:

Sample Input	Sample Output

9
12 13 15 14 16 17 18 19 20
3
1 5 4
2 4 2
4 9 6
	15
	14
	20

Explanation:

The second query asks you to find the 2nd value from the **N** input values, starting from the 2nd index value of the input to the 4th index value. The sorted values on this index range are 13, 14 15. The 2nd value of the sorted values is 14, which is the answer to the second query.



Problem Setter: Abu Quwsar Ohi, Intake 33, Dept. of CSE.

J. Sorting a Binary Array

Input: Standard Input, **Output:** Standard Output
Time Limit: 1 second(s)
Memory Limit: 256 megabytes

Problem Statement:

Given a binary array that means each element of that array is either 0 or 1. Your task is to sort the array using minimum swaps. You are allowed to swap only adjacent elements.

Input:

The input contains multiple test cases. First-line contains an integer **T** $(1 \leq \mathbf{T} \leq 10)$. The first line of every test case contains an integer **N** $(1 \leq \mathbf{N} \leq 2 * 10^5)$. Then the next line contains **N** elements of that array.

Output:

For each test case, output minimum numbers of swap required to sort the binary array.

Sample Input/Output:

Sample Input	Sample Output

2
5
0 1 0 1 0
8
0 0 1 0 1 0 1 1
	3
	3