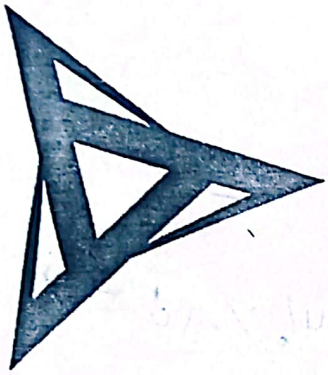
Welcome to PUCon'25

"First, solve the problem. Then, write the code."

– John Johnson



PF-PUCon'25

Final Round

Programming Competition

FCIT

University of the Punjab

Problem A

Some N theory!

A positive integer N is a 400 number if and only if it satisfies both of the following two conditions:

1. N has exactly 2 distinct prime factors.
2. For each prime factor p of N, p divides N an even number of times. More formally, the maximum non negative integer k such that p^k divides N is even.

You are given Q queries. Each query gives you an integer A, and you must find the **largest 400 number not exceeding A**.

It is guaranteed that under the constraints of this problem; a 400 number not exceeding A always exists.

Input:
- The first line contains an integer Q — the number of queries ($1 \le Q \le 2 \times 10^5$).
- Each of the next Q lines contains a single integer A ($36 \le A \le 10^{12}$).

Output:
For each query, print the largest 400 number not exceeding A.

Constraints:
- $1 \le Q \le 2 \times 10^5$
- $36 \le A \le 10^{12}$
- All input values are integers

Sample Input:
```
5
404
36
60
1000000000000
123456789
```

Sampe Output:
```
400
36
36
1000000000000
123454321
```

Problem B

Festival Stall Arrangement

You are organizing a festival, and you have n stalls to decorate. You have three types of decorations available A, B, and C. The cost of decorating each stall with a specific decoration is different.
You need to decorate all the stalls such that no two adjacent stalls have the same decoration. Your goal is to find the minimum total cost to decorate all the stalls while ensuring no adjacent stalls have the same decoration.
The costs for decorating each stall with each decoration type are given in an **n x 3** cost matrix, where costs[i][j] represents the cost of decorating stall i with decoration type j.

- 0 for decoration A
- 1 for decoration B
- 2 for decoration C

Input:
An integer t ($1 \le t \le 100$) indicating the number of test cases.
For each test case:

- A line containing two integers n and m, where n is the number of stalls and m = 3 (the number of decorations).
- Followed by n lines, each containing m space-separated integers.
  Each integer costs[i][j] ($0 \le costs[i][j] \le 5000$) represents the cost of decorating stall i with decoration j.

Output:
For each test case, return an integer representing the minimum total cost to decorate all stalls such that no two adjacent stalls have the same decoration. Print each result on a new line.

Sample Input:

```
2
3 3
10 2 15
3 7 9
1 5 6
2 3
0 1 7 2
1 0 1 9 4
```

Sample Output

```
10
3
```

Explanation:

- Test Case 1: Decorate stall 0 with decoration B (cost 2), stall 1 with decoration A (cost 3), and stall 2 with decoration B (cost 5). The minimum total cost is 2 + 3 + 5 = 10.
- Test Case 2: Decorate stall 0 with decoration C (cost 2), and stall 1 with decoration A (cost 1). The minimum total cost is 2 + 1 = 3.

Problem C

Hackathon

The Department of Computer Science at **FCIT University** is organizing a hackathon. The participants are seated in a circular arrangement, represented by a binary circular array nums. Each student is marked with a 1 if they belong to the programming club, and a 0 otherwise.

The programming club members (marked as 1) need to be seated together to form a single group. However, since the seats are arranged in a circle, the first and last seats are adjacent, making it a **circular array.**

To prepare for the event efficiently, the organizers want to know the **minimum number of swaps** needed to group all programming club members (1s) together in any location in the circle.

A swap is defined as taking two distinct positions in an array and swapping the values in them.

A circular array is defined as an array where we consider the first element and the last element to be adjacent.

Given a binary circular array nums, return *the minimum number of swaps required to group all 1's present in the array together at any location.*

Input Format:
The first line contains the number of test cases. The 2nd line contains space separated array

Output Format:
Minimum number of swaps.

Sample Input:

```
3
0 1 0 1 1 0 0
0 1 1 1 0 0 1 1 0
1 1 0 0 1
```

Sample Output:

```
1
2
0
```

Test Case 1 Explanation:
Here are a few of the ways to group all the 1's together:
[0,0,1,1,1,0,0] using 1 swap.
[0,1,1,1,0,0,0] using 1 swap.
[1,1,0,0,0,0,1] using 2 swaps (using the circular property of the array).
There is no way to group all 1's together with 0 swaps.
Thus, the minimum number of swaps required is 1.

Test Case 2 Explanation:
Here are a few of the ways to group all the 1's together:
[1,1,1,0,0,0,0,1,1] using 2 swaps (using the circular property of the array).
[1,1,1,1,1,0,0,0,0] using 2 swaps.
There is no way to group all 1's together with 0 or 1 swaps.
Thus, the minimum number of swaps required is 2.

Test Case 3 Explanation:
All the 1's are already grouped together due to the circular property of the array.
Thus, the minimum number of swaps required is 0.

Problem D

Labyrinth of Codes

Beneath the hallowed halls of FCIT lies an ancient labyrinth – a matrix filled with mysterious "codes." Legend has it that only the wisest adventurers can navigate the labyrinth by collecting a sequence of codes that unlock its secrets. The challenge is that in each row of the labyrinth, exactly one code must be chosen, and as you move downward, the column indices of your selected codes must strictly increase. With the right selection, the total power of the collected codes is maximized, granting safe passage out of the labyrinth. However, if the labyrinth is too narrow, it might be impossible to escape.

You must guide the adventurers through the labyrinth, which is represented as a grid of **N rows** and **M columns**, where each cell contains an integer representing the power of a code. To unlock the secrets of the labyrinth and find the path with the **maximum total power**, the adventurers must follow these rules:

- They must collect exactly one code from each row.
- If they choose a code from column **j** in row **i**, the next chosen code (from row **i+1**) must come from a column with an index **strictly greater than j**.

If the labyrinth is too narrow – meaning there are more rows than columns ($N > M$) – then escape is impossible, and the adventurers must accept failure, returning **-1**.

Input Format

The first line contains an integer T – the number of labyrinths (test cases). For each labyrinth:

- The first line contains two integers N and M – the number of rows and columns.
- The next N lines each contain M integers, where the j-th integer of the i-th row represents the power of the code at that position in the labyrinth.

Output Format

For each test case, output a single line containing the maximum power the adventurers can collect or **-1** if escape is impossible.

Constraints

- $1 \le T \le 10$
- $1 \le N, M \le 500$
- $-10^9 \le$ Code power $\le 10^9$

Sample Input

```
2
3 3
1 2 3
4 5 6
7 8 9
4 3
1 2 3
4 5 6
7 8 9
10 11 12
```

Sample Output

```
15
-1
```

Labyrinth 1: The adventurers can take the path $1 \rightarrow 5 \rightarrow 9$, collecting a total power of **15**.

Labyrinth 2: With 4 rows and only 3 columns, there isn't enough space for strictly increasing column indices, making escape impossible. Hence, the output is **-1**.

Problem E

Dr. Doom vs. Fantastic Four

In the Marvel Universe, the sinister Dr. Doom is scheming to defeat the Fantastic Four with his latest invention: a Cosmic Node Array powered by n unique energy nodes, each labeled with a distinct integer from 1 to n, forming a permutation. These nodes are arranged in a specific order, and Doom has discovered that certain pairs of nodes can synchronize to unleash a Cosmic Surge, amplifying his weapon's power to crush his enemies.

Each node is placed at a position indexed from 1 to **n**. A pair of nodes at positions i and j (where $1 \le i < j \le n$) generates a Cosmic Surge if the product of their labels ($p_i \times p_j$) is divisible by the product of their position numbers ($i \times j$) without remainder. The Fantastic Four led by Reed Richards (Mr. Fantastic) need your help to analyze Doom's node arrays and count how many pairs produce these dangerous surges to foil his evil plan.

Your mission is to calculate, for each of Dr. Doom's node arrays, the number of node pairs that generate a Cosmic Surge.

Input:
The first line contains a single integer t ($1 \le t \le 10^4$), representing the number of test configurations. Then, the description of each configuration follows.
For each configuration:

- The first line contains a single integer n ($1 \le n \le 5 \times 10^5$), the number of energy nodes in the array.
- The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$), representing the labels of the nodes in their arranged order (a permutation of the numbers 1 to n).

It is guaranteed that the sum of n across all configurations does not exceed 5×10^5.
Output:
For each configuration, output a single integer: the number of node pairs ($1 \le i < j \le n$) such that the product of the node labels ($p_i \times p_j$) is divisible by the product of their position numbers ($i \times j$) without remainder.

Sample Input 1

```
3
3
1 2 3
5
1 2 4 3 5
3
2 3 1
```

Sample Output 1

```
0
2
3
```

Sample Input 2

```
2
3
2 3 1
12
8 9 7 12 1 10 6 3 2 4 11 5
```

Sample Output 2

```
1
9
```

Problem F

Queries for Zero Array

Time Limit: 2 seconds
Memory Limit: 256 MB

You are given an integer array a of length n. You are also given m queries, where the i-th query is represented as a pair [li, ri] (0-indexed). Each query allows you to **decrement each element in the subarray a[li ... ri] by at most 1**. That means, for each position j in the range [li, ri], you can choose to decrease a[j] by either 0 or 1 (independently for each position).
You are allowed to **remove any number of queries** (including zero). However, after removing some queries, you must ensure that it is **still possible to convert the array a into a zero array** — that is, an array where every element is equal to 0 — by applying the remaining queries in some order.
Your task is to find the **maximum number of queries you can remove** such that it is still possible to convert the array into a zero array using the remaining queries.
If it is not possible to convert the array into a zero array even when using **all** the queries, you must output -1.

Constraints:

- $1 <= nums.length <= 10^5$
- $0 <= nums[i] <= 10^5$
- $1 <= queries.length <= 10^5$
- $queries[i].length == 2$
- $0 <= l_i <= r_i < nums.length$

Input
The first line contains two integers n and m — the length of the array and the number of queries.
The second line contains n integers a[0], a[1], ..., a[n-1] — the initial array.
Each of the next m lines contains two integers li and ri ($0 <= li <= ri < n$) — the indices representing the i-th query.
Output
Print a single integer — the **maximum number of queries you can remove** such that the array can still be converted into a zero array using the remaining queries. If it is not possible to convert the array to all zeros even using all the queries, print -1.

Sample Input 1:
3 3
2 0 2
0 2
0 2
1 1
Sample Output 1:
1

Sample Input 2:
4 4
1 1 1 1
1 3
0 2
1 3
1 2
Sample Output 2:
2

Sample Input 3:
4 1
1 2 3 4
0 3
Sample Output 3:
-1

Problem G

Large Hadron Collider

The "Large Hadron Collider" (LHC) has particles coming in and out of existance all the time. The scientists at the LHC think they would be able to discover new particles if only the collider was larger. Hence the plan was approved to build a new larger collider called "The Very Large Hadron Collider" (VLHC).

Before the project starts the architects need some information. Specifically they need to know what will be the maximum number of particles that will be inside the collider at any single time instance. You are given the arrival and leaving times of n particles in the VLHC. Your task is to find the maximum number of particles that were in the VLHC at any time?

Input Format:

The first input line has an integer **n**: the number of particles.

After this, there are **n** lines that describe the particles. Each line has two integers **a** and **b**: the arrival and leaving times of a particle.

You may assume that all arrival and leaving times are distinct.

Output Format:

Print one integer: the maximum number of particles.

Constraints:

Time limit: 1.00 s Memory limit: 512 MB

$1 <= n <= 2 * 10^5$

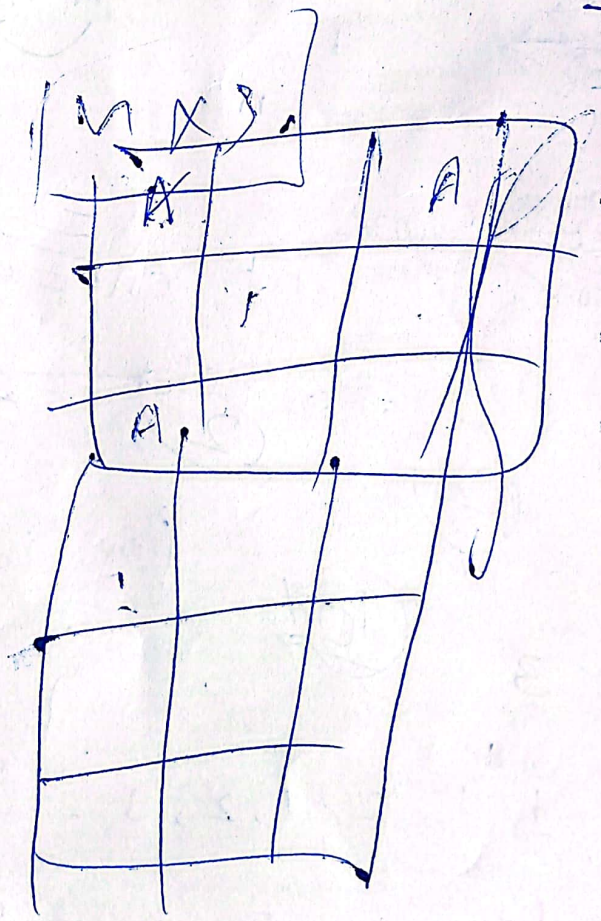
$1 <= a < b <= 10^9$

Sample Input:

```
10
1 20
2 19
3 18
4 17
5 16
6 15
7 14
8 13
9 12
10 11
```

Sample Output:

```
10
```



Problem H

Mystic Staircase Sequence

You have discovered a mysterious staircase in the ruins of an ancient temple. Each step you climb reveals a sequence of glowing runes. The first step reveals **1**, the second step reveals **1 2**, the third step reveals **1 2 3**, the fourth step **1 2 3 4**, and so on. The staircase seems infinite, and the sequence of runes continues as:

CopyEdit
1, 1 2, 1 2 3, 1 2 3 4, 1 2 3 4 5, ...

This gives rise to a long, single sequence of runes like:
1, 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5, ...

You are tasked with discovering **which rune appears at the n-th position** in this sequence.

Input
The first line contains, single integer t (1<=t<=100), representing the number of test cases.
The second line contains, single integer n ($1 \le n \le 10^{14}$) — the position of the rune you are interested in.

Output
Print the value of the rune at the n-th position in the sequence (1-indexed).

Test_Case#1:

Input:
4
3
5
55
56

Output:
2
2
10
1