NEUB ICT Fest 2025
Programming Contest

Hosted by

CodePanja

Contest Schedule & Rules

Contest Schedule

Duration

3 hours

Standings Frozen

Last 30 minutes

Judge Rules

Supported Languages

You can use C17 GCC 14.1, C++20 GCC 14.1, Java 17.0.6, Python 3.8.1, Golang 1.23.5, Ruby 2.7.0, Bash 5.0.0, PHP 8.3.11 and Rust 1.85.0 languages in this test.

Penalty System

If the problem has a penalty points then the penalty for each non-accepted submission, measured in minutes, is added to the contestant's total standing time only if the problem is eventually solved. For problems left unsolved, no penalty time will be applied.

Output Comparison

CodePanja automatically normalizes output by removing all trailing whitespaces and trailing newlines from both your program's output and the expected output before comparison. This means you don't need to worry about extra spaces or newlines at the end of your output.

Java Code Requirements

The Java code class name must be Main. Check the following correct code:

```
import java.util.Scanner;

// The class name must be Main
public class Main {
  public static void main(String[] args) {
    // sample code
    Scanner in = new Scanner(System.in); // Standard input
    System.out.println(in.nextInt() + in.nextInt()); // Standard output
  }
}
```

ONLINE_JUDGE Macro

The ONLINE_JUDGE macro is unsupported on CodePanja. The below code won't work:

```
#ifndef ONLINE_JUDGE
// for getting input from input.txt
freopen("input1.txt", "r", stdin);
// for writing output to output.txt
freopen("output1.txt", "w", stdout);
#endif
```

A Welcome to NEUB

Time Limit: 1 sec Memory Limit: 256 MB Penalty: No penalty

Welcome to the **NEUB ICT Fest 2025**! You are going to be a great programmer one day, and this is (likely) the first step in your journey to become one. Let's start with an easy problem as your first onsite contest experience.

Your task is simple: given a positive integer **n**, determine whether it is **odd** or **even**.

Input

The input consists of a single line containing a positive integer n.

Constraints

In this section, problem-setters usually specify the size or scope of the input values. Sometimes, they just don't provide any information. In this problem, **we also don't want to tell you anything about the constraints**. Get prepared for any worst-case scenario—just like the real world, which is always messy!

Output

Print odd if n is odd. Print even otherwise.

Sample Data

Sample Input 1

```
5
```

Sample Output 1

```
odd
```

Sample Input 2

```
22
```

Sample Output 2

```
even
```

B Array Extraction Queries

Time Limit: 1 sec Memory Limit: 512 MB Penalty: No penalty

You are given multiple independent scenarios.
In each scenario, you are provided with an integer N representing the length of a sequence and an integer Q representing the number of operations to perform.

You are then given a sequence of N integers.
For each operation, a single integer X is given.

Your task is to determine the **largest possible combined value** that can be formed by selecting **exactly X elements** from the sequence, without altering their relative order requirement (you may skip elements if necessary).

You must answer all Q operations for every scenario independently.

Subtask #1 (10 points):

$1 \leq N, Q \leq 100$

Subtask #2 (30 points):
$1 \leq N, Q \leq 10^4$

Subtask #3 (60 points):
$1 \leq N, Q \leq 10^5$

Input

The first line contains a single integer T representing the number of test cases.

Each test case contains the following:

- The first line contains two integers N and Q.

- The second line contains N integers $A_1, A_2, A_3, \ldots, A_N$

- The next Q lines each contain a single integer X.

Output

For each operation in every scenario, output a single integer:
The **maximum achievable combined value** obtained by selecting exactly X elements from the given sequence.

Each result should be printed on a new line, following the order of the operations as they appear in the input.

If there are multiple scenarios, their outputs should appear **consecutively** in the same order as the scenarios are provided.

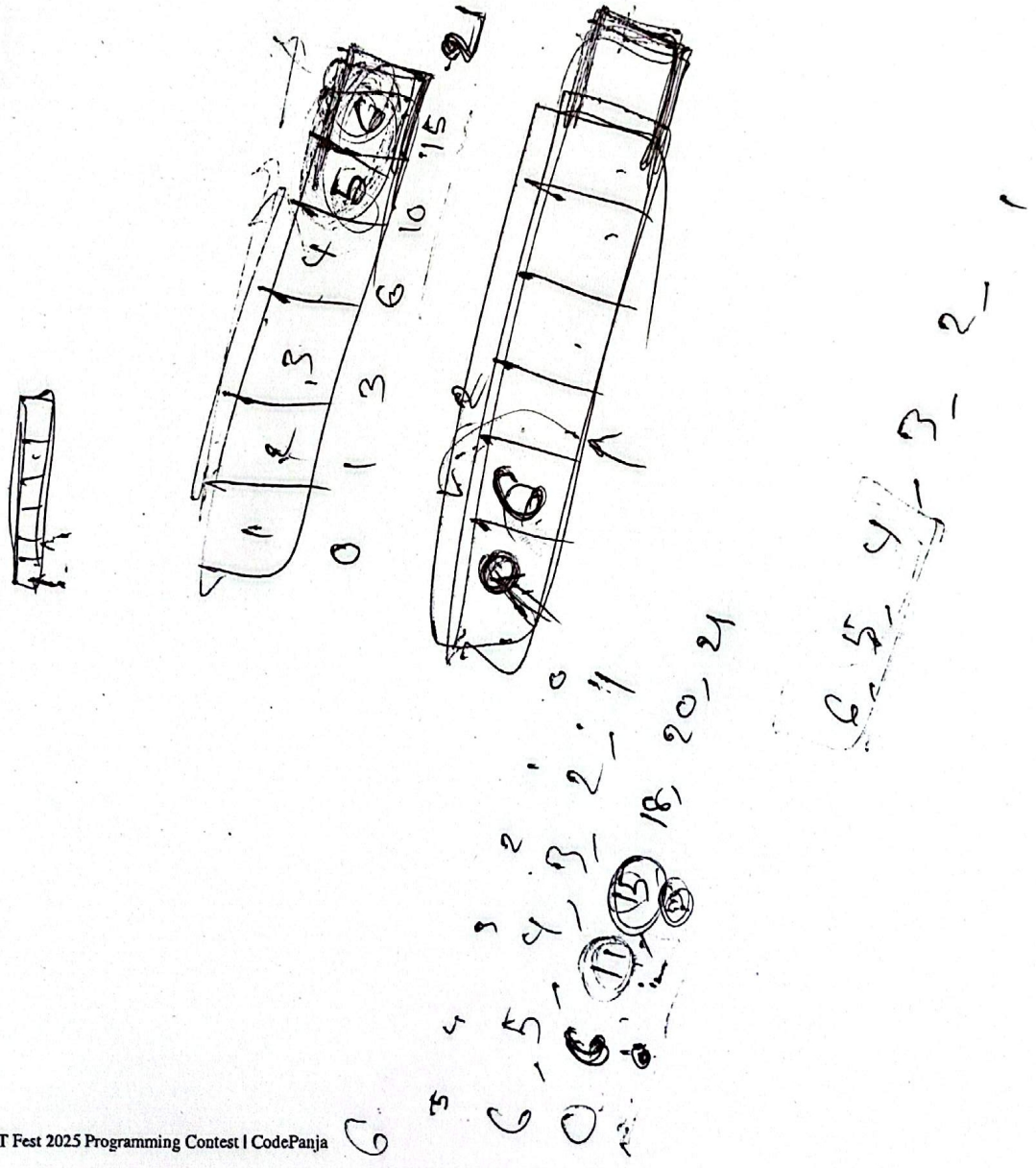
Sample Data

Sample Input 1

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

Sample Output 1

```
7
18
```



C Nine-Eleven

Time Limit: 1 sec Memory Limit: 256 MB Penalty: No penalty

In the heart of SeriousLand, there lies a park shaped as an 11×11 grid — totaling 121 cells. On a cloudy evening, 9 people were peacefully enjoying their time in the park.

Suddenly, a lightning bolt struck the area, igniting fire in some parts of the park. The fire spreads aggressively — every unit of time, it expands to the four adjacent cells: up, down, left, and right. That means if a cell $[i, j]$ is on fire at time T, then the cells $[i+1, j]$, $[i-1, j]$, $[i, j+1]$, and $[i, j-1]$ will catch fire at time $T+1$.

The screams caught the attention of SeriousLand's only firefighter — Thakur. Equipped with a drone that can instantly drop him onto any cell in the park, Thakur immediately arrives at the scene. However, the drone can only transport Thakur himself, not the unconscious victims.

To rescue someone, Thakur must land near them, lift the unconscious person onto his shoulders, and carry them out of the park by foot, exiting the grid through any of its borders. While carrying someone, Thakur can move one cell at a time in any of the four directions (up, down, left, or right), but only through cells that are not yet consumed by fire, and each move costs one unit of time.

Unfortunately, saving all 9 people may not be possible. So, Thakur aims to rescue as many as he can before the fire reaches them. Use your programming skills for the greater good—help Thakur determine the maximum number of lives he can save.

Input

11×11 grid of characters.

The character 'F' denotes fire, 'P' denotes person, and '#' denotes an empty cell.

Output

One integer: the maximum number of lives Thakur can save.

Sample Data

Sample Input 1

```
# # # # P # F # # # #
# # # # # F # P # # #
# # # # # # # # # # #
# # # # # # # # # P #
P # # # # F # P # # #
F # # F # F # P # # #
# # # # # # # # # # #
# # # # # # # # # # #
P # # # # # # # # # #
# # # # P # # # P # #
# # # F # # # # # # #
```

Sample Output 1

```
5
```

Notes

Thakur will save people from cell (5,1) (1,5) (9,1) (4,10) (10,9) and can not save others from fire.

D The Drone Defense System

Time Limit: 1 sec Memory Limit: 256 MB Penalty: No penalty

At **North East University Bangladesh (NEUB)**, the CSE department has launched an experimental **autonomous drone defense system** to guard the campus.

Each academic building is connected by **straight paths** where students walk, represented as **line segments** on a 2D map.

A **security drone** hovers over the **central courtyard**, maintaining a **circular detection zone**. Any moving object (like a person or vehicle) entering this circular zone will instantly trigger an alert to the NEUB control center.

Your task is to **analyze the campus layout** and determine **which walking paths** enter or touch the drone's detection radius. If a path even slightly grazes the detection boundary, it must raise an **"ALERT"** signal.

You are given:

- The **coordinates** of the drone's position $C(x, y)$.
- The **radius** r of the detection zone.
- n **path segments (straight lines)**, each defined by two endpoints $A(x_{i1}, y_{i1})$ and $B(x_{i2}, y_{i2})$.

Input

x y r

n

x_{11} y_{11} x_{12} y_{12}

x_{21} y_{21} x_{22} y_{22}

... ...

x_{n1} y_{n1} x_{n2} y_{n2}

- **(x, y)**: drone's position. [$-10^4 \le x, y \le 10^4$]
- **r**: detection radius. [$1 \le r \le 10^4$]
- **n**: number of walking paths. [$1 \le n \le 10^5$]
- **Next n lines**: coordinates of each path's two endpoints. [$-10^4 \le x_{i1}, y_{i1}, x_{i2}, y_{i2} \le 10^4$]

All coordinates are **integers**.

Output

For each path, print

- "ALERT" → if the path touches or crosses the detection zone.
- "CLEAR" → if the path stays completely outside.

For all calculations involving floating-point numbers, an absolute difference of 10^{-9} or less should be treated as **equality**. When comparing two real numbers, a and b, they are considered equal if $|a - b| \leq 10^{-9}$.

Sample Data

Sample Input 1

```
0 0 6
4
-10 0 10 0
0 7 8 7
-5 -5 2 2
6 6 9 9
```

Sample Output 1

```
ALERT
CLEAR
ALERT
CLEAR
```

E Parity Swap Chaos

Time Limit: 1 sec Memory Limit: 256 MB Penalty: No penalty

Alice's teacher gave her an array of n integers as a homework problem. She can swap two adjacent elements of the array if and only if they have different parity.

In other words, you can swap elements in positions $(i, i+1)$ only if $a_i \bmod 2 \neq a_{i+1} \bmod 2$.

After trying many times, Alice got tired and asked for your help to find the **lexicographically smallest** array possible.

⇒ *sequence a is lexicographically smaller than a sequence b if there exists an index i such that $a_j = b_j$ for all $j < i$, and $a_i < b_i$.*

Input

The first line contains a single integer t ($1 \leq t \leq 10^4$), the number of test cases.

For each test case:

The first line contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$), the length of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers, the lexicographically smallest array that can be obtained.

Sample Data

Sample Input 1

```
2
5
1 2 3 4 5
4
2 4 6 8
```

Sample Output 1

```
1 2 3 4 5
2 4 6 8
```

Notes

In the first test case, Alice can swap adjacent elements with different parity to eventually arrange the array as [1, 2, 3, 4, 5].

In the second test case, all elements are even, so no swaps are possible. The answer remains [2, 4, 6, 8].

F Abdul Karim and the Rickshaw Race

Time Limit: 1 sec Memory Limit: 256 MB Penalty: No penalty

In the busy town of Sylhet Sadar, there's an annual Rickshaw Race organized by Abdul Karim, the legendary rickshaw puller. There are N checkpoints along the main road — and Karim must visit all of them in order. Each checkpoint gives him a certain amount of energy (+ve means energy gain, −ve means he gets tired). Karim starts the race with zero energy and must always keep his energy non-negative throughout the race — otherwise, he faints and loses! You can choose where he starts (at which checkpoint), but once he starts, he must go in order and wrap around circularly until he's back at the starting checkpoint. Your job is to find the smallest index (1-based) checkpoint from which Karim can start the race and finish it successfully without ever having negative energy. If it's impossible for him to finish the full circle, print -1.

Input

The first line contains a single integer N— the number of checkpoints.

The second line contains N integers $E_1, E_2, \ldots, E_N$— the energy gained or lost at each checkpoint.

Constraints:

$1 \leq N \leq 10^5$

$-10^4 \leq E_i \leq 10^4$

Output

Print one integer — the index of the checkpoint where Abdul Karim should start the race, or -1 if it's impossible.

Sample Data

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

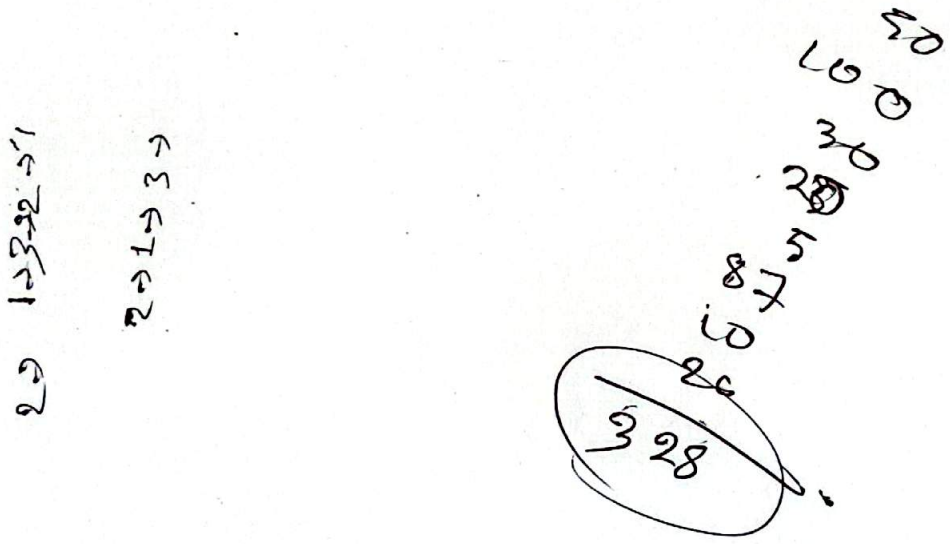
```
4
-1 2 -2 1
```

Sample Output 2

```
2
```

Notes

Explanation Of Example 1: If Karim starts at checkpoint 3: energy = 2 → 1 → 3 → 2 → 5 He never goes below 0, so 3 is the best starting point



G Max Cost Production

Time Limit: 2 sec Memory Limit: 256 MB Penalty: No penalty

You're the manager of a special factory that produces a unique product. The factory has n machines, and each machine i can produce items with value a_i.

The final product is created by selecting some machines and multiplying their output values together. Due to system design, if you use k machines to create a product, the efficiency cost is k^2.

More machines means better efficiency a higher k^2 value indicates a better quality product!

You have the ability to modify at most one machine's value to any positive integer of your choice before production. The modified machine's value must be > 1.

Given a target value X, determine the maximum efficiency cost k^2 needed to produce a product with value exactly X. If it's impossible to produce exactly X (even after the optional modification), output -1.

Input

The first line contains two integers n and X ($1 \leq n \leq 10^5$, $2 \leq X \leq 10^{12}$) the number of machines and the target value.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($2 \leq a_i \leq 10^{12}$) the values of each machine.

Output

Print a single integer the maximum efficiency cost k^2 to produce exactly X, or -1 if it's impossible.

Sample Data

Sample Input 1

```
5 32
2 2 2 2 2
```

Sample Output 1

```
25
```

Sample Input 2

```
3 12
4 5 2
```

Sample Output 2

```
4
```

Notes

In the first example:

$n = 5$, $X = 32$, and $a = [2, 2, 2, 2, 2]$.

Using all machines gives $2 \times 2 \times 2 \times 2 \times 2 = 32 = X$,

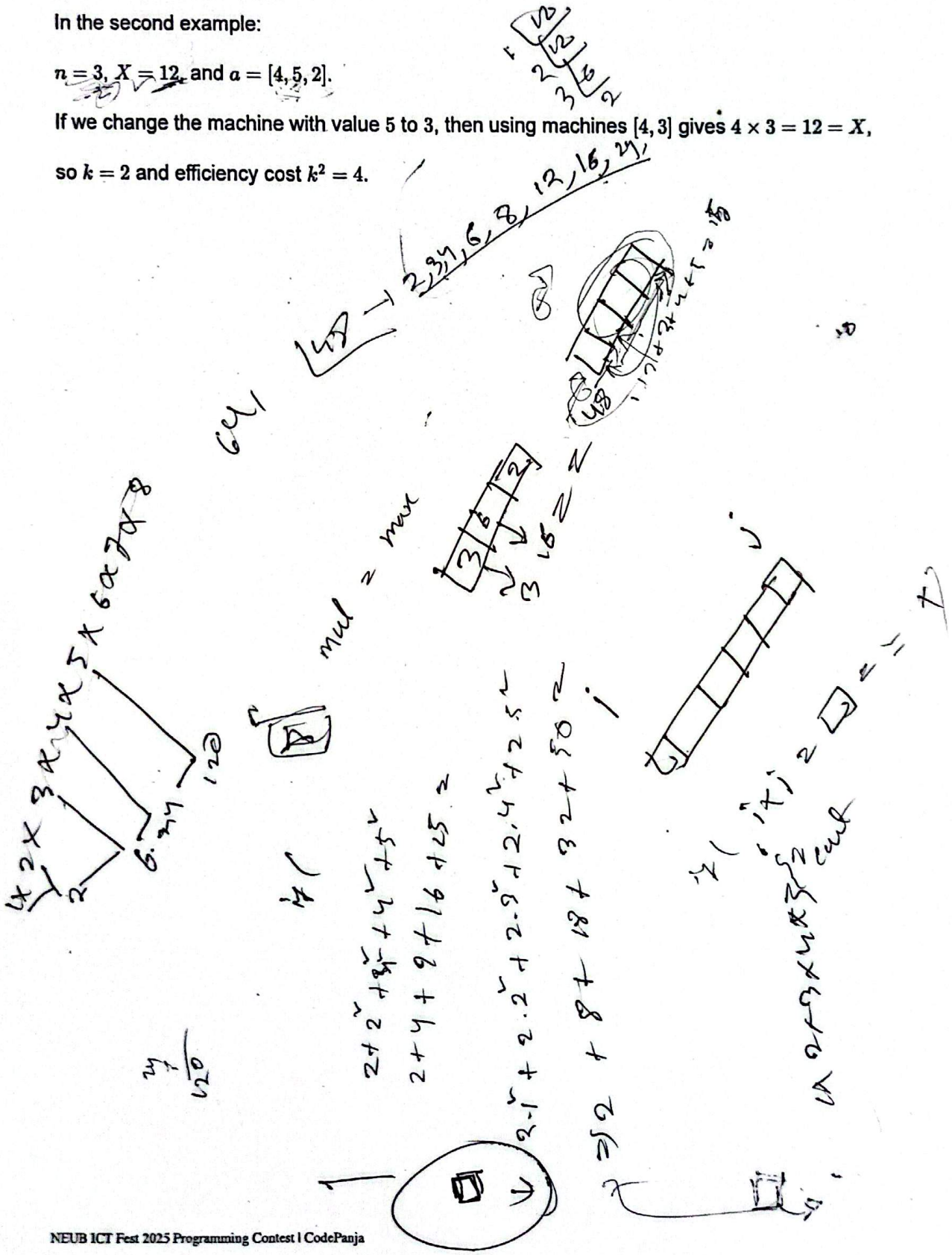
so $k = 5$ and efficiency cost $k^2 = 25$.

In the second example:

$n = 3$, $X = 12$, and $a = [4, 5, 2]$.

If we change the machine with value 5 to 3, then using machines $[4, 3]$ gives $4 \times 3 = 12 = X$,

so $k = 2$ and efficiency cost $k^2 = 4$.



H Cloudy Days

Time Limit: 1 sec Memory Limit: 256 MB Penalty: No penalty

Alice is navigating the country of Cloudland on a cloudy day, making it challenging to see her way. She has a car with K units of fuel remaining, where each unit of fuel allows her to travel a distance of 1 unit.

Given the country's layout, consisting of several interconnected cities, and her current location, your task is to determine and print the total possible number of cities she can visit if she exhausts all her fuel.

Input

The input starts with four integers:

- N: The number of cities in Cloudland.
- M: The number of roads connecting different cities in Cloudland.
- K: The remaining fuel units in Alice's car.
- C: Alice's current city.

The following M lines describe the road connections between cities as pairs of integers u, v, indicating a bidirectional road between cities u and v.

Output

Output a single integer representing the total possible number of cities Alice can reach if she uses up all K units of her fuel.

Constraints:

$2 \leq N \leq 10^3$

$0 \leq M \leq (N \times (N-1))/2$

$1 \leq K \leq 10^6$

$1 \leq C \leq N$

Sample Data

Sample Input 1

```
6 5 2 1
2 5
1 5
3 4
2 4
5 3
```

Sample Output 1

```
4
```

Notes

1. Alice starts in city 1 with 2 units of fuel.
2. She can move to city 5 using 1 unit of fuel (remaining fuel = 1).
    - From city 5, she can reach city 3 using 1 unit of fuel (remaining fuel = 0).
3. Alternatively, from city 5, she can reach city 2 using 1 unit of fuel (remaining fuel = 0).

At this point, Alice has 0 units of fuel and has reached cities 3 and 2. There are no further cities she can reach with her remaining fuel. The total number of cities she can reach is {1, 5, 3, 2} through all possible paths. So, the answer for the sample input, considering all possible paths, is 4.