Think.
Create.
Solve.

2022 Asia-Manila Regional
Programming Contest
14–15 December 2022
Ateneo de Manila University

Problem Set

ATENEO DE MANILA UNIVERSITY

icpc international collegiate programming contest
icpc.foundation

JET BRAINS

HUAWEI

EXPEDOCK

Smart

Contents

Notes

- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use BufferedReader and PrintWriter.
  - In C/C++, use scanf and printf.
- All problems are solvable in C++, and Java, but the same is not guaranteed for Python due to its slowness.
- Good luck and enjoy the problems!

Very important! Read the following:

- Your solution will be checked by running it against several hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.
- The output checker is **very strict**. Follow these guidelines strictly:
  - It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
  - It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
  - Do not print any tabs. (No tabs will be required in the output.)
  - Do not output anything else aside from what's asked for in the Output section. So, do not print things like "Please enter t".

  Not following the output format strictly and exactly will likely result in a **Wrong answer** verdict.
- Do not read from, or write to, a file. You must read from the standard input and write to the standard output.
- For Java, do not add a package line at the top of your code. Otherwise, your submission will be judged **Runtime Error**.
- Only include one file when submitting: the source code (.cpp, .java, .py, etc.) and nothing else.
- The file containing your submitted source code may only have a size of at most 256 KB.
- Only use letters, digits and underscores in your filename. Do not use spaces, or other special symbols.
- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use BufferedReader and PrintWriter.
  - In C/C++, use scanf and printf.

  We recommend learning and using these functions during the Practice Session.
- All problems are solvable in C++, and Java, but the same is not guaranteed for Python due to its slowness.
- Good luck and enjoy the contest!

Problem A

An Easy Calculus Question

Time Limit: 1 second

The *International Calculus Problem Committee* has generously donated a problem for today's contest. Let's all say thank you to them!

Let a, b, c, and d be real constants. Now, let the piecewise function f be defined on all real numbers x as follows:

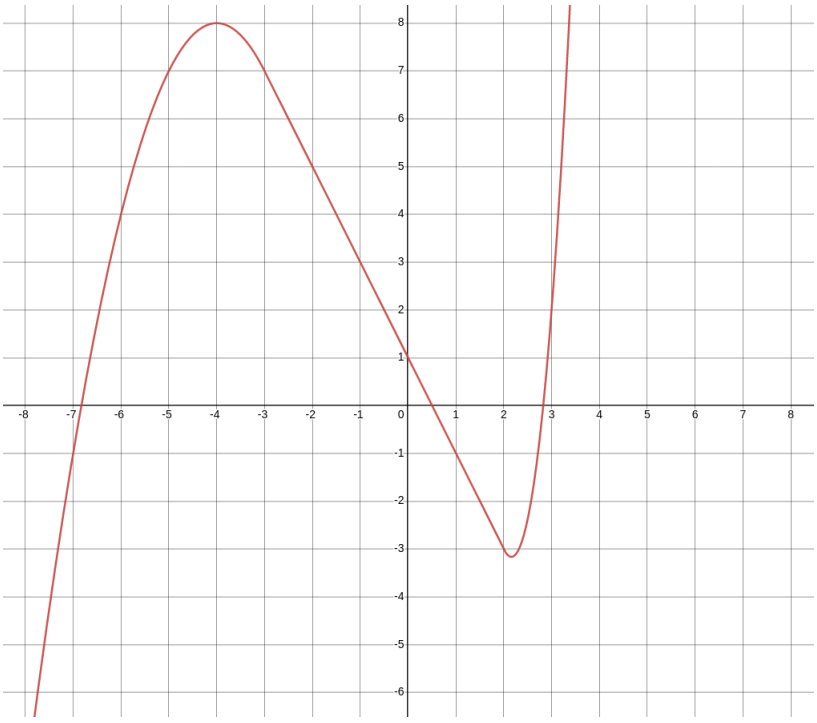
$$f(x) = \begin{cases} -(x+4)^2 + 8, & \text{if } x \in (-\infty, -3], \\ ax + b, & \text{if } x \in (-3, 2], \\ x^3 + cx + d, & \text{if } x \in (2, +\infty). \end{cases}$$

Furthermore, you are told that f is *differentiable everywhere*. We can show that there actually exists exactly one set of values for a, b, c, and d for which f is differentiable everywhere, and therefore this condition is sufficient to uniquely determine f.

This is the task: Given some integer x as input, output the value of $f(x)$. It is possible to show that if x is an integer, then the value of $f(x)$ will also always be an integer.

Actually, wait a minute, the Chief Judge says that this question isn't appropriate because this is a programming competition, not a calculus class.

So, to make things fair, we'll also give you a **HINT**: f is differentiable everywhere if and only if $a = -2$, $b = 1$, $c = -14$, and $d = 17$. *These are the values that you should use for f.* Look, I'll even show you the graph of the function in order to prove that it's differentiable everywhere.



The graph of $y = f(x)$ if we take $a = -2$, $b = 1$, $c = -14$, and $d = 17$. We can see that the function is differentiable everywhere. This image was generated using Desmos.

Input Format

Input consists of only a single line containing the integer x.

Constraints

- $-10 \leq x \leq 10$

Note: Constraints are *guaranteed* to be true. You don't have to check them; you may simply assume them to be true.

Output Format

Output a single integer, the value of f when evaluated at the given x.

We remind you to **not** output the decimal point or any places after the decimal point.

Sample Input 1	Sample Output 1
−7	−1

Sample Input 2	Sample Output 2
0	1

Sample Input 3	Sample Output 3
3	2

Explanation

In the third sample input, we wish to evaluate $f(3)$. By the definition of the piecewise function, because $3 \in (2, +\infty)$, we are going to evaluate $x^3 + cx + d$ at $x = 3$. Using the values of c and d given in the hint, we arrive at the value of,

$$3^3 + (-14)3 + 17 = 27 - 42 + 17 = 2.$$

Problem B

Better than Bitcoin

Time Limit: 1.5 seconds

Alice and Bob have decided to invent a new cryptocurrency based on prime numbers, called the *Internet's Coolest PrimeCoin*. This is an absolutely fabulous idea, and there is no way that it could possibly go wrong.

As a reminder, a prime number is an integer *greater than* 1 that has no divisors other than 1 and itself. *In modern mathematics,* 1 *is not considered a prime number.*

After several months' worth of computations and shameless abuse of *the cloud*, Alice and Bob have finally managed to compute the first n prime numbers. These prime numbers shall serve as the basis of the cryptocurrency's Proof of Work... somehow. We don't know the details.

Anyway! Now we must decide who gets to keep each of these first n primes. For each prime, you must either give it to Alice or give it to Bob. Don't throw any away; they worked hard for those! We want to distribute the primes such that the total value that Alice and Bob each receive is *proportional* to the effort that each one contributed to the project.

Let A be the sum of the primes given to Alice, and let B be the sum of the primes given to Bob. Alice and Bob decided that it would be fair to divvy up the total value in a ratio of $p : q$. Thus, some distribution of primes will be considered *fair* if $A : B = p : q$. Because they're feeling extra cute today, p **and** q **are also guaranteed to be prime numbers**.

Given n, p, and q, *count* the number of different ways to fairly distribute the first n prime numbers between Alice and Bob. Two distributions are considered different if there was a prime number that was given to Alice in one, but to Bob in the other. The answer can be huge, so we only ask you to output the answer modulo 1169996969. Also, there will be T independent test cases per file.

Input Format

Input begins with a line containing the positive integer T.

Then, T lines follow. Each line contains the three space-separated integers n, p, and q, meaning that for this test case, the first n prime numbers must be distributed between Alice and Bob with total values in a $p : q$ ratio.

Constraints

- $1 \leq T \leq 10^5$
- $2 \leq n \leq 2000$
- $2 \leq p, q \leq 30$
- p and q are prime

Output Format

For each test case, output a single integer, the number of ways to fairly distribute the first n prime numbers between Alice and Bob. Since these answers can be quite large, you only need to output them modulo 1169996969.

Sample Input 1	Sample Output 1
2 3 7 7 8 2 5	2 4

Sample Input 2	Sample Output 2
3 1859 19 7 1967 2 17 2000 29 29	213519321 1086566377 0

Explanation

The first three prime numbers are $\{2,3,5\}$. To distribute these with a $7:7$ ratio, we can give $\{2,3\}$ to Alice and $\{5\}$ to Bob, *or* we can give $\{5\}$ to Alice and $\{2,3\}$ to Bob.

The first eight prime numbers are $\{2,3,5,7,11,13,17,19\}$. To distribute these with a $2:5$ ratio, we can...

- Give $\{3,19\}$ to Alice and $\{2,5,7,11,13,17\}$ to Bob.
- Give $\{5,17\}$ to Alice and $\{2,3,7,11,13,19\}$ to Bob.
- Give $\{2,3,17\}$ to Alice and $\{5,7,11,13,19\}$ to Bob.
- Give $\{2,7,13\}$ to Alice and $\{3,5,11,17,19\}$ to Bob.

Problem C
Conform Conforme
Time Limit: 2 seconds

The *Illustrious College Promoting Conformity*, Manila Campus, has one central theme which its mission and vision both revolve around: to teach its students that life is better when you're exactly the same as everyone else.

Fun Fact: Clinical studies have shown that expressing your individuality causes other people to perceive you as abrasive, irritating, obnoxious, and just an all-around awful human being.

The college has n students, labeled 1 to n, each with a shirt that displays any positive integer of their choosing. Initially, before the first day of school, we know that a_i is the number on student i's shirt.

Fun Fact: If you're exactly like everyone else, then other people will perceive you as safe, secure, comforting, and familiar, which are all desirable traits for human interaction.

The school's culture swiftly instills in them a desire to form cliques of people that are exactly like them. At the end of each of day, each student **simultaneously** does the following: If the number on their shirt is v, then they replace it with $\text{freq}(v)$. Here, $\text{freq}(v)$ counts the number of people *that day* whose shirt displayed the value v (this includes themselves in the count).

Fun Fact: If you don't have any friends, maybe try completely changing everything about your personality in order to make yourself more likeable!

Trends are dynamic, and the students are always constantly reassessing what's popular, ready to change the number on their shirt (and their entire personality) at the end of every day. It's getting a bit hard to keep track of!

As a favor to the administration of the college, you must answer the following question: Given the initial values on their shirts and some integer k, what will be the value on each student's shirt at the end of the kth day?

Input Format

The first line of input contains the space-separated integers n and k.

The second line of input contains the n space-separated integers $a_1, a_2, a_3, \ldots, a_n$.

Constraints

- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq k \leq 10^9$
- $1 \leq a_i \leq 10^9$ for each i

Output Format

Output n space-separated integers, the values on the shirts of the students (in order) at the end of the kth day.

Sample Input 1	Sample Output 1
8 1 2 7 1 8 2 8 1 8	2 1 2 3 2 3 2 3

Sample Input 2	Sample Output 2
7 2 6 7 1 1 1 9 9	2 2 3 3 3 2 2

Explanation

In the first sample input, initially,

- There are $\text{freq}(1) = 2$ occurrences of the number 1.
- There are $\text{freq}(2) = 2$ occurrences of the number 2.
- There are $\text{freq}(7) = 1$ occurrences of the number 7.
- There are $\text{freq}(8) = 3$ occurrences of the number 8.

Thus, for example, all students with 8 on their shirt will change that number to 3 at the end of the first day.

In the second sample input, note that there is only one student wearing the shirt with the number 6, so they replace it with the number 1 at the end of the first day; similarly, the student wearing a 7 also changes their number to 1. In general, the values on the shirts are $\{1, 1, 3, 3, 3, 2, 2\}$

But then the next day, those two note that there are *two* students now wearing a 1 on their shirt, so at the end of the second day, both of them change their numbers to 2.

Problem D
Domination Devil
Time Limit: 6 seconds

Today, the Domination Devil is going to teach us how to subjugate an island nation! You just need to remember this: *Instability Causes Pure Chaos*.

This country has n different islands, ranked from 1 to n in terms increasing order of *power*. Initially, there existed $n(n-1)/2$ bidirectional bridges, with each bridge directly connecting two different islands together.

We need to cripple this nation's ability to protect itself, but we need to do so in such a way that their economy is *technically* still intact. We're going to choose a subset of the bridges, and *destroy* every bridge in this chosen subset. When you destroy a bridge, it is no longer usable.

After destroying the chosen bridges, both of these conditions must be met:

- *Destroy each island's support structure.* Each island should only be directly connected to **at most** k other islands with a greater power ranking than it. Two islands are directly connected if a usable bridge exists that connects them.
- *Leave them with just the bare minimum needed to function, hanging only by a thread.*. It should still be possible to visit any island from any other island, just by traveling through the remaining usable bridges.

Given n and k, count the number of different subsets of bridges, such that destroying all the bridges in this subset will satisfy both of these conditions. This number can be quite large, so we only ask you to output the answer modulo 1699741697.

Input Format

Input consists of a single line with the space-separated integers n and k.

Constraints

- $1 \leq k < n \leq 2 \cdot 10^5$

Output Format

Output a single integer, the number of subsets that satisfy both of the stated conditions, modulo 1699741697.

Sample Input 1	Sample Output 1
4 1	6

Sample Input 2	Sample Output 2
4 2	30

Sample Input 3	Sample Output 3
200000 1	1581480553

Sample Input 4	Sample Output 4
200000 100000	1212906613

Explanation

Let (u, v) denote the bridge connecting islands ranked u and v. If $n = 4$, then the set of all bridges is $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$. With $k = 1$, the following bridges-to-destroy subsets will result in both given conditions being satisfied:

- Destroy $(1,3)$, $(1,4)$, and $(2,3)$
- Destroy $(1,2)$, $(1,4)$, and $(2,3)$
- Destroy $(1,2)$, $(1,3)$, and $(2,3)$
- Destroy $(1,3)$, $(1,4)$, and $(2,4)$
- Destroy $(1,2)$, $(1,4)$, and $(2,4)$
- Destroy $(1,2)$, $(1,3)$, and $(2,4)$

Problem E

Escape from Markov

Time Limit: 2.5 seconds

A crazed warlock has harnessed the power of the Koschei in order to resurrect a famous Russian mathematician as an undead lich. The mathematician, an expert in probability theory (specifically, stochastic processes), used his expertise in order to optimize the operations of the warlock's private military, allowing them to violently seize control of the country.

You are a member of a group of rebels whose sworn purpose is to overthrow the warlock-turned-warlord and restore freedom to the land. If you can destroy the phylactery which houses the lich's soul, then the mathematician will become mortal once more. Following the death of his grand logistician, you expect that the warlock's regime will crumble, and your country will be liberated once more. This is: Escape from Markov

There are n cities in the country, numbered 1 to n. These cities are connected by m bidirectional roads. Crossing any of these roads in either direction always takes exactly 1 hour. No road connects a city to itself, and any pair of cities is connected by at most one road. Travel between cities can *only* be done using these roads, and it is possible to reach any city from any other city using only these roads. You are currently located at city a. You need to reach city b as soon as possible, since you know that Markov's soul is stored somewhere there.

Unfortunately, the warlock has p patrol cars (formally known as the *Insurgent-Catching Police Cars*) which monitor the roads. Each patrol car has its own patrol plan, which graph theorists would describe as a closed circuit.

Each patrol car has a list of l cities that they visit, in order; when they are done visiting their lth city, they return to their starting city, and then start the patrol over. Currently, each patrol car is located at the first city in their list.

Any two consecutive cities in each plan (including the lth and first cities) are directly connected by a road. That is because patrol cars can also only travel between cities by taking the same roads you do. Note that a patrol car may visit the same cities or roads multiple times in its patrol plan. It also takes each of them exactly 1 hour to traverse each road in either direction.

While a patrol car is out of the city and on some road, that road becomes impossible to cross. If you try to cross a road while any patrol car is *anywhere* on that road (no matter what direction you or the car are headed) *besides* the city endpoints, you are caught, and the rebellion is crushed. However, you can also wait and hide indefinitely at any city for as long as you want without getting caught (even if a patrol passes through the city you are in at that time).

Given these conditions, find the shortest amount of time (in hours) that it will take you to get from city a to city b without getting caught, or report if this task is impossible.

Input Format

The first line of input contains four space-separated integers n, m, p, and l.

Then, m lines follow, each containing two space-separated integers u and v, meaning that a bidirectional road exists that connects cities u and v. No road connects a city to itself, and any pair of cities is connected by at most one road.

Then, p lines follow, each corresponding to the patrol plan of some patrol car. Each line will contain l space-separated integers. The first integer is the patrol car's starting city, and the remaining sequence describes the cities that the patrol car then visits, in order. Recall that after visiting the lth city in its plan, the patrol car returns to the first city and then starts its patrol over.

The last line of input contains two space-separated integers a and b.

Constraints

- $2 \leq n \leq 2 \cdot 10^5$
- $n - 1 \leq m \leq 2 \cdot 10^5$
- $1 \leq p \leq 2 \cdot 10^5$
- $2 \leq l \leq 2 \cdot 10^5$
- $p \cdot l \leq 10^6$
- You can visit any city from any other city by only using the roads.
- There exists a road connecting any two consecutive cities in a patrol plan
- $a \neq b$
- All cities in the input are between 1 and n inclusive

Output Format

Output a single integer, the shortest amount of time (in hours) it will take for you to go from city a to city b, or the word `IMPOSSIBLE` if the task is impossible.

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
2
```

Sample Input 3	Sample Output 3
6 5 1 4 1 2 2 3 3 4 4 5 5 6 5 6 5 4 1 6	7

Sample Input 4	Sample Output 4
2 1 1 2 1 2 2 1 1 2	IMPOSSIBLE

Explanation

For the first sample input, we start at city 1. Note that for the first hour, there is a patrol car headed from city 2 to city 1, so we can't head that way. We can avoid the patrol car by going from city 1 to 4, and then from city 4 to 3, taking a total of 2 hours.

For the second sample input, we would want to head to city 2 directly, but there is a patrol car monitoring the road between cities 1 and 2 for the first hour. So we wait for 1 hour at city 1. After the hour has passed and the patrol car has moved on, we proceed directly from city 1 to city 2. Overall, it took us a total of 2 hours.

Problem F

Factions vs The Hegemon

Time Limit: 1.5 seconds

For some reason I can't explain
I know Saint Peter won't call my name
Never an honest word
But that was when I ruled the world.

Viva la Vida, Coldplay

Long ago, there existed an ancient civilization whose records have mostly been lost to history. It is known that there were n warring factions, arranged and labeled 1 to n in order from west to east. The *wealth value* of the ith warring faction is summarized in a single positive integer w_i.

The warring factions would never achieve true peace with one another, but it was known that they would collude and form truces in order to conspire against what they believed to be the largest threat to them all. Of course, this only matters if their combined might can actually take down said largest threat.

We say that this civilization is in *hegemony* if there exists a faction whose wealth value is strictly greater than the combined wealth values of all *other* factions.

According to records, the following process occurred $n - 1$ times throughout history.

- One of the factions *collapses*.
    - If this civilization is not in hegemony, then the faction with the *largest* wealth value collapses.
    - If this civilization *is* in hegemony, then the faction with the *smallest* wealth value collapses.
    - In case of a tie, the west-most most/least wealthy faction is the one that collapses.
- After this, the nearest non-collapsed factions to its west and east each (if they exist) have their wealth values increased by half the wealth value of the collapsed faction, rounded down.
    - Even if the collapsed faction bordered only one other faction, that other faction *still* only receives half the collapsed faction's wealth value. The other half is lost to, I don't know, barbarians or something.

After $n - 1$ iterations of this, only one faction remains. But even then, this civilization fell to, like, I guess the Mongols? Or maybe the Romans. Either way, the records are lost to time.

But luckily, as a computer scientist, you can just simulate this process and find the exact order in which the factions collapsed, and how much wealth each had when they did.

Input Format

The first line of input contains a single integer n.

The second line of input contains n space-separated integers $w_1, w_2, w_3, \ldots, w_n$, the initial wealth values of the factions.

Constraints

- $2 \leq n \leq 2 \cdot 10^5$
- $1 \leq w_i \leq 10^9$ for each i

Output Format

Output n lines, describing each faction in the order that they collapsed (first to collapse comes first). Each faction should be described by two space-separated integers: its label, and its wealth value at the moment of collapse.

Sample Input 1	Sample Output 1
5 3 1 4 9 1	4 9 3 8 1 3 2 6 5 12

Sample Input 2	Sample Output 2
6 12 4 12 1 1 7	1 12 3 12 5 1 4 7 6 10 2 24

Explanation

Here are the events that occur in the second sample input.

- The wealthiest factions have wealth value 12, and of such factions, faction 1 is the west-most one. The civilization is not in hegemony, so faction 1 collapses. Half its wealth goes to faction 2. Factions $2, 3, 4, 5, 6$ now have wealth values $\{10, 12, 1, 1, 7\}$.
- Faction 3, with wealth 12, collapses because the civilization is not in hegemony. Factions 2 and 4 each receive half its wealth value. Factions $2, 4, 5, 6$ now have wealth values $\{16, 7, 1, 7\}$.
- Faction 2 is the wealthiest, and with a wealth value of 16, the civilization is now in hegemony. The least wealthy faction, 5, collapses. Note that $1/2$, rounded down, is 0, so the factions nearest to 5 gain 0 wealth. Factions $2, 4, 6$ now have wealth values $\{16, 7, 7\}$.
- We are still in hegemony, and faction 4 is the west-most faction whose wealth value is the minimum of 7. It collapses, and factions 2 and 6 now have wealth values $\{19, 10\}$.
- We are still in hegemony, and faction 6, with a wealth of 10, is now the weakest one, so it collapses. Faction 2 is all alone with a wealth value of $\{24\}$, until it too collapses.

Problem G
Gallivanting Merchant

Time Limit: 2 seconds

I Can Program C++! is a new edutainment video game that has been crowdfunded by the competitive programming community. Its goal is to teach children aged 3 to 4 all about the wonders of segmentation faults, memory leaks, and move semantics!

The developers have chosen a Fantasy RPG for their game's setting. You take on the role of a *Programming Wizard* who defeats dragons and tyrants and the embodiment of evil *by using the power of programming*! Wow! That sounds so fun!

Each day, you can choose a different activity to do around town. This includes special events that only happen on certain days, so we number the in-game days $1, 2, 3, 4, \ldots$.

One such special event is the appearance of a merchant that sells rare, exclusive items that will aid you on your programming journey, like RGB keyboards, gamer chairs, and brain cells.

This merchant is only available on special days, according to the following rules: You get to *choose* on what day the merchant first appears, and from that point on, the merchant appears regularly once every k days. Formally, the value of k is fixed, but you can ask the merchant to first appear on some day s, which would mean that the merchant appears on days s, $s+k$, $s+2k$, $s+3k$, $\ldots$.

The merchant's wares *also* change depending on the day! The merchant has n different items for sale, each of which is only sold during some interval of time. The ith item is sold only on days L_i through R_i (inclusive). Of course, you can only purchase an item if the merchant is available during any of the days during which the item is being sold.

You are given k, and the descriptions of each of the n items. If you are optimal in your choice of s (the day the merchant first arrives), what is the maximum number of *different* items that you can acquire from the merchant?

Input Format

The first line of input contains the space-separated integers n and k.

Then, n lines follow, where the ith of these contains the space-separated integers L_i and R_i.

Constraints

- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq k \leq 10^9$
- $1 \leq L_i \leq R_i \leq 10^9$ for each i

Output Format

Output a single integer, the maximum number of different items that you can acquire from the merchant.

Sample Input 1	Sample Output 1
3 5 2 6 6 11 16 21	3

Sample Input 2	Sample Output 2
8 4 2 4 9 10 1 2 4 5 5 7 2 10 11 11 11 13	6

Sample Input 3	Sample Output 3
4 100 100 101 101 102 102 103 103 104	2

Problem H

HIIT

Time Limit: 1 second

Alright contestants, I want you to drop right now and give me 10 burpees! 20 pushups! 30 squats! 40 Supermans! And then I want you running high knees and butt kicks for 50 seconds! And then I want you to do all of that all over again! Thrice!

You didn't think that ICPC would just be a mental competition, did you? No way! If your own body isn't operating at peak capacity, then how can you expect your computer to be as well? This routine isn't even that hard, you know!

Alright, now, I want 1 minute each of planks and mountain climbers, and if you don't know the proper form for those exercises, just read the documentation! When you're done with that... we start with the programming problem! It's like chess boxing!

Alice drafted up an exercise routine for Bob to follow, which she calls the *Intense Cardio Punisher Challenge*. It consists of n different exercises, but since Bob is just a beginner, each exercise has an *easy version* which consumes a_i units of energy, and an *intense version* which consumes b_i units of energy. For each exercise, Bob must determine if he wants to do the easy version, or if he wants to do the intense version, or if he wants to skip that exercise today.

Bob can expend up to x units of energy today; any more than that and he, uh, *dies*. Primarily, Alice is happiest when Bob does not die. As a secondary priority, Alice is happiest when Bob skips as few exercises as possible. As a tertiary priority, Alice is then happiest when Bob does as many *intense* exercises as possible.

Bob is too tired to think right now, so let's help him decide on a plan that will make Alice as happy as possible!

Input Format

The first line of input contains the space-separated integers n and x.

The second line of input contains the n space-separated integers $a_1, a_2, \ldots, a_n$, the energy costs of the easy versions of the exercises.

The third line of input contains the n space-separated integers $b_1, b_2, \ldots, b_n$, the energy costs of the intense versions of the exercises.

Constraints

- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq a_i < b_i \leq 10^9$ for each i
- $1 \leq x \leq 10^{15}$

Output Format

Output a string of n characters, encoding what Bob should do for each exercise.

- If the ith character is `0`, then Bob skips the ith exercise (consuming 0 units of energy).
- If the ith character is `1`, then Bob does the easy version of the ith exercise (consuming a_i units of energy).
- If the ith character is `2`, then Bob does the intense version of the ith exercise (consuming b_i units of energy).

Your solution will be accepted if all of the following are satisfied:

- The total expended energy should be $\leq x$
- Among all solutions that use $\leq x$ energy, the number of `0`s should be minimized.
- Among all solutions that use $\leq x$ energy and have the minimal number of `0`s among such solutions, the number of `2`s should be maximized.

If there are multiple possible solutions, any will be accepted.

Sample Input 1

```
4 6
1 5 2 1
3 7 3 4
```

Sample Output 1

```
1021
```

Sample Input 2

```
5 44
14 11 12 15 8
15 18 17 18 16
```

Sample Output 2

```
20021
```

Sample Input 3

```
5 15
1 2 3 4 5
2 3 4 5 6
```

Sample Output 3

```
11111
```

Sample Input 4

```
5 99
100 101 102 103 104
105 106 107 108 109
```

Sample Output 4

```
00000
```

Explanation

For the first sample input, `2011` would also have been an acceptable solution.

The second sample input has many other acceptable solutions aside from the one given.

Problem I
Item Crafting
Time Limit: 2 seconds

The *Item-Crafting Prowess Competition* is a fun event that players of MyCraft can challenge each other to, in order to flex their knowledge of the game's rich crafting system! There are m different items in MyCraft, with internal ID numbers 1 to m. Only some of these items can be found in the overworld; let's call them *raw materials*.

Every other item has to be made using the game's crafting system. Each item that is not a raw material has some *recipe*, which is some list of other items. In order to craft this item, you have to consume **one** of each item listed in its recipe.

It may then be possible that this product could be included as an ingredient in the recipe of some other item. That's what makes this crafting system so rich and multi-layered! In order to ensure there are no impossible recipes, you are guaranteed that all ingredients in the recipe of some item will always have a *greater* ID number than that of the final product.

The first n items are guaranteed to not be included in the recipes of any other items. Let's call these the *legendary final products*. The goal of the game is to craft *as many possible different legendary final products as possible* (i.e. making the same legendary final product multiple times will only count once towards your score).

Suppose you were able to acquire some number of raw materials, and no other items. What is the maximum number of different types of legendary final products that can be made?

Input Format

The first line of input contains two space-separated integers n and m.

The following m lines each represent an item. The ith line, describing the item with internal ID i, will start with an integer c_i, followed by a space.

- If $c_i = 0$, then this item is a raw material. A single integer p_i follows, the amount of this material that you have.
- Otherwise, if $c_i > 0$, then c_i space-separated integers follow, listing the IDs of the ingredients in the recipe for this item.

The legendary final products will always just be the first n items.

Constraints

- $1 \leq n \leq 15$
- There are at most 10 raw materials (items with $c_i = 0$).
- $n < m \leq 2 \cdot 10^5$
- $0 \leq c_i \leq 2 \cdot 10^5$, and the sum of c_i over all items i will not exceed $5 \cdot 10^5$.
- $0 \leq p_i \leq 10^8$ for each raw material
- $c_i > 0$ for each legendary final product (i.e. if $1 \leq i \leq n$)

- Each recipe consists of distinct ingredients.
- For the ith item's recipe (if it has one), the IDs of all its ingredients are strictly greater than i and n.
- All IDs are between 1 and m (inclusive)

Output Format

Output a single integer, the maximum number of different final products that can be made.

Sample Input 1

```
2 6
2 3 4
3 4 5 6
0 1
0 1
0 1
0 1
```

Sample Output 1

```
1
```

Sample Input 2

```
2 6
2 3 4
3 4 5 6
0 1
0 2
0 1
0 1
```

Sample Output 2

```
2
```

Sample Input 3

```
1 5
2 2 3
2 3 4
2 4 5
0 3
0 2
```

Sample Output 3

```
1
```

Sample Input 4

```
1 4
3 2 3 4
0 2
1 4
0 1
```

Sample Output 4

```
0
```

Explanation

In the first sample test case, there are two final products: one requires items 3 and 4 in its recipe, and the other one requires items 4 and 5 and 6 in its recipe. We can make one or the other, but not both, therefore the answer is 1.

The second sample test case is similar to the first one, except we have two of item 4. This allows us to make both legendary final products, and so the answer is 2.

In the third sample test, we need to do the following:

- Make two of item 3.
- Make one of item 2. This consumes the remaining item 4, as well as one of the item 3 that we just made.
- We have one of item 2 and one of item 3, so we can use them to make item 1.

Problem J

Junior Steiner Three

Time Limit: 2 seconds

The Pacific Ocean has had it too good for too long. Look at that huge expanse of nothing but water... what a smug jerk. The *Inter-Continental Pacific Connection* is a project spearheaded by Team Magma, whose aim is to add more glorious land to this world!

Let's model the Pacific Ocean as a grid that has been subdivided into r rows and c columns. Each cell in this grid is either **land** or **water**. Humans live on the land, which makes land awesome, and they drown when they enter the water, which makes water lame. A human can walk between two land cells if and only if they share an edge (sharing a corner is not enough).

Team Magma can pay 1 million pesos in order to transform any water cell of their choice into a land cell. They would like to transform the grid such that starting from any land cell, it is possible to reach any other land cell just by walking.

In a perfect world, they would be able to accomplish this goal by just transforming the *entire* Pacific Ocean into land. But this is not a perfect world. In order to respect budget constraints, please help Team Magma figure out how to achieve their goal using the minimum possible cost.

...Okay, so maybe this problem is too difficult. Fine, to make things easier, let's limit ourselves to this specific case: In the given grid, **exactly three** of its cells are land.

Input Format

The first line of input contains the space-separated integers r and c.

Then, r lines of input follow, each containing a string of length c. This encodes the state of the cells in the grid.

- The period `.` character means that the corresponding cell has water
- The hash `#` character means that the corresponding cell has land

Constraints

- $2 \leq r, c \leq 100$
- There will always be **exactly** three `#` characters in the grid, with the rest being `.` characters.

Output Format

Output r lines that each contain a string of c characters, corresponding to the grid from the input *after* you have transformed some of its cells into land.

Your solution will be accepted if it satisfies all of the following:

- The land cells from the input are still land cells in the output
- Starting from any land cell, it should be possible to reach any other land cell in the grid just by walking.

- Among all solutions that satisfy the two previous requirements, your answer should also *minimize* the number of cells which were transformed from water into land.

If there are multiple possible solutions, any will be accepted.

Sample Input 1

```
4 5
.....
..#..
....#
.#...
```

Sample Output 1

```
.....
..#..
..###
.##..
```

Sample Input 2

```
3 3
..#
.#.
#..
```

Sample Output 2

```
.##
##.
#..
```

Sample Input 3

```
4 3
..#
.##
...
...
```

Sample Output 3

```
..#
.##
...
...
```

Problem K

Kapitan Amazing

Time Limit: 1 second

Captain Amazing (*Kapitan Kamangha-mangha* in the original Tagalog version) is investigating the mysterious disappearances of several prominent superheroes, and his detective-work has led him to infiltrating the secret lair of an as-of-now unknown supervillain. Most people don't think that he and the other "dumb brute" supers are capable of smart detective work, but Captain Amazing likes to prove them wrong. He calls it the *Investigative Capabilities of Powerhouse Capes*

He was able to find the room that contains the data that he needs, but unfortunately, the room is protected by password. The input device for the password is a standard QWERTY keyboard whose layout can be described with three strings, as follows:

```
QWERTYUIOP
ASDFGHJKL
ZXCVBNM
```

Captain observed that the security guards here do not wear any gloves, meaning that they leave trace amounts of oil on everything they touch. Also, it seems that this keyboard is only used for inputting the password, and nothing else. Therefore, Captain concludes that the keys with a significant amount of oil residue on them *and only those keys* must be included in the password!

For example, consider the following QWERTY keyboard. The letters whose keys had a significant amount of oil residue on them have been replaced with `*` asterisks.

```
QWERTYU*O*
*SDFGHJK*
ZX*VB**
```

This reveals that the password contains the letters `I`, `P`, `A`, `L`, `C`, `M`, `N`, and *only* those letters.

We can deduce that none of the following are possible passwords:

- `CLAMPING`, because otherwise the letter `G` would also have had oil residue.
- `MAILMAN`, because otherwise the letters `P` and `C` would *not* have had oil residue.
- `PASSWORD`, for many reasons.

On the other hand, these are all possible passwords:

- `ICPCMANILA`
- `CLIPMAN`
- `CAMPANILLA`
- `ALPACAMANIA`
- `IPAAAAAAAAAAAAAAAAAAAAAAAALCMN`

Let's help Captain Amazing! Given the oily keyboard, your program must then be able to answer Q different queries of the following form: Given some string s, is s a possible password?

Input Format

Input begins with three lines, describing a QWERTY keyboard in the same format as described in the problem statement. Some of these letters have been replaced by asterisks $\star$, meaning that the keys with those letters had oily residue.

This is followed by a line containing a single integer Q, the number of queries to answer.

Then, Q lines follow, each containing a string s.

Constraints

- At least one key is oily
- $1 \leq Q \leq 100$
- Each s consists of at most 30 uppercase letters

Output Format

For each query, output `POSSIBLE` if its s is a possible password, considering the oily keyboard, and `IMPOSSIBLE` if it is not a possible password.

Sample Input 1

```
QWERTYU*O*
*SDFGHJK*
ZX*VB**
8
CLAMPING
MAILMAN
PASSWORD
ICPCMANILA
CLIPMAN
CAMPANILLA
ALPACAMANIA
IPAAAAAAAAAAAAAAAAALCMN
```

Sample Output 1

```
IMPOSSIBLE
IMPOSSIBLE
IMPOSSIBLE
POSSIBLE
POSSIBLE
POSSIBLE
POSSIBLE
POSSIBLE
```

Sample Input 2

```
QWERTYUIOP
*SDFGHJKL
ZXCVBNM
5
A
AA
AAAA
AAAAAAAA
AAAAAAAAHHHHHHHH
```

Sample Output 2

```
POSSIBLE
POSSIBLE
POSSIBLE
POSSIBLE
IMPOSSIBLE
```

Problem L

LCG Manipulation

Time Limit: 2 seconds

A linear congruential generator (LCG) is a process that yields an infinite stream of pseudo-random numbers. This process is used by many video games in order to quickly generate a sequence of "good enough" random numbers for various non-deterministic processes.

For example, consider the Pokémon games. Whenever you catch a new Pokémon, each of its six stats receives an integer bonus "uniformly randomly" selected from $+0$ to $+31$. The intent is to model how each individual Pokémon has its own strengths and weaknesses that differentiate it from others of the same species. These bonus stats are decided by the output of an LCG.

Competitive players need their team to be at peak strength. However, the chances of landing a perfect $+31$ on *all* six stats is $1/2^{30}$, or a little less than one in one *billion*. But what if I told you that, *somehow*, a knowledgeable player can "miraculously" get perfect-statted Pokémon in only a handful of attempts.

In fact, in many games, it is possible to rig seemingly-random in-game events in order to always produce the optimal outcome. How!? Well, let's delve a little bit into how LCGs work...

An LCG generates an infinite sequence $x_0, x_1, x_2, x_3, \ldots$ which is determined by four values: the non-negative integers a, b, s, and p. Let $x_0 = s$ (which is why s is often called the "seed"). For each $n \geq 1$, define the next term in the sequence as,

$$x_n = (ax_{n-1} + b) \bmod p$$

In general, the modulus can be any number, but for this problem, **we use the letter p because we are only considering the cases where the modulus is prime**.

The LCG may *seem* random, but it's actually quite predictable! If you can uncover the values of a, b, s, and p, and if you know that the optimal outcome occurs when the LCG generates a value of v, then you can use a computer program to know exactly how long you would need to wait until your game is in the desired "random" state.

I think you can guess the problem now. You must perform an *Implausible Chance's Probability Calculation.* Given the values of a, b, s, p, and some value v, find the minimum non-negative integer n such that $x_n = v$; if no such n exists, then you should report so as well. Also, there will be T independent test cases per file.

Input Format

Input begins with a line containing the positive integer T, the number of test cases.

Then, T lines follow. Each line contains the five space-separated integers a, b, s, p, and v.

Constraints

- $1 \leq T \leq 40$
- $2^3 - 1 \leq p \leq 2^{31} - 1$
- p is prime

- $0 < a, b < p$
- $0 \leq s, v < p$

Output Format

For each test case, output a single line:

- If an n exists such that $x_n = v$, then output the minimum such n
- Otherwise, output the word `IMPOSSIBLE`

Sample Input 1

```
8
2 3 1 7 5
2 3 1 7 6
2 3 1 7 1
2 3 1 7 0
7 1 2 31 24
7 1 2 31 25
7 1 2 31 26
7 1 1 31 26
```

Sample Output 1

```
1
2
0
IMPOSSIBLE
6
4
IMPOSSIBLE
2
```

Sample Input 2

```
2
314159265 1234567890 2022 2147483647 2023
1234567890 314159265 2022 2147483647 2023
```

Sample Output 2

```
1058977885
IMPOSSIBLE
```

Explanation

Let's suppose that $(a, b, s, p) = (7, 1, 2, 31)$.

Then, we would have $x_0 = 2$ and $x_n = (7x_{n-1} + 1) \bmod 31$.

- $x_1 = (7 \times 2 + 1) \bmod 31 = 15$.
- $x_2 = (7 \times 15 + 1) \bmod 31 = 13$.
- $x_3 = (7 \times 13 + 1) \bmod 31 = 30$.
- $x_4 = (7 \times 30 + 1) \bmod 31 = 25$.
- $x_5 = (7 \times 25 + 1) \bmod 31 = 21$.
- $x_6 = (7 \times 21 + 1) \bmod 31 = 24$.
- $\vdots$