IEEEXTREME PROGRAMMING COMPETITION 13.0

October 19, 2019

Table of Contents

Ranged Alfa Pool

Time limit: *2480 ms*
Memory limit: *470 MB*

In Alfa Pool, players play against each other in pairs, in the order dictated by the system. For each game, the winner earns a given number of points and the loser earns no points.

To make the tournament more interesting, the organizers decided on the following set of rules:

- The points awarded are doubled for every successive win. The first win earns 1 point, the second successive win earns 2 points, the third successive win earns 4 points, and so on.
- In case of a loss, the successive win streak resets. A subsequent win earns again 1 point.
- If a player loses twice in a row, they are eliminated from the tournament.

Find out in how many different ways a player can earn between A and B points, before being eliminated. For example, let $A = 2$ and $B = 4$. A player can earn between 2 and 4 points in 12 different ways, where a number K denotes a win earning K points and **X** denotes a loss.

```
1 X 1 X X
X 1 X 1 X X
1 X 1 X 1 X X
X 1 X 1 X 1 X X
1 X 1 X 1 X 1 X X
X 1 X 1 X 1 X 1 X X
1 X 1 2 X X
X 1 X 1 2 X X
1 2 X X
X 1 2 X X
1 2 X 1 X X
X 1 2 X 1 X X

```

In all the 12 scenarios above, the player exited the tournament with a total of either 2, 3, or 4 points.

Standard input

Your program must read from the standard input. The first line contains the number of queries N that you have to answer. Each of the following N lines contains one query, consisting of two space-separated non-negative integers A_i and B_i.

Standard output

Your program must print to the standard output exactly N lines, each containing exactly one integer number: the number of different ways in which a player can earn between A_i and B_i points before exiting the tournament. For each query, you have to print the result modulo $10^9 + 7$.

Constraints and notes

- $1 \le N \le 10^4$
- $0 \le A_i \le B_i \le 10^6$

Input	Output	Explanation
1 2 4	12	This is the example shown above.

Alfa Pool

Time limit: *1280 ms*

Memory limit: *370 MB*

In an online pool tournament, players play against each other in pairs, in the order dictated by the system. For each game, the winner earns a given number of points and the loser earns no points.

To make the tournament more interesting, the organizers decided on the following set of rules:

- The points awarded are doubled for every successive win. The first win earns 1 point, the second successive win earns 2 points, the third successive win earns 4 points, and so on.
- In case of a loss, the successive win streak resets. A subsequent win earns again 1 point.
- If a player loses twice in a row, they are eliminated from the tournament.

Find out in how many different ways a player can earn B points, before being eliminated. For example, let $B = 5$. A player can earn 5 points in 8 different ways, where a number K denotes a win earning K points and **X** denotes a loss.

```
1 X 1 X 1 X 1 X 1 X X
X 1 X 1 X 1 X 1 X 1 X X
1 2 X 1 X 1 X X
X 1 2 X 1 X 1 X X
1 X 1 2 X 1 X X
X 1 X 1 2 X 1 X X
1 X 1 X 1 2 X X
X 1 X 1 X 1 2 X X

```

In all the 8 scenarios above, the player exited the tournament with a total of 5 points.

Standard input

Your program must read from the standard input. The first line contains the number of queries N that you have to answer. Each of the following N lines contains one query, consisting of one non-negative integer B_i.

Standard output

Your program must print to the standard output exactly N lines, each containing exactly one integer number: the number of different ways in which a player can earn B_i points before exiting the tournament. For each query, you have to print the result modulo $10^9 + 7$.

Constraints and notes

- $1 \leq N \leq 10^4$
- $0 \leq B_i \leq 10^5$

Input	Output	Explanation
1 5	8	This is the example shown above.

The Lonely Shepherd

Time limit: *1280 ms*
Memory limit: *264 MB*

Bogdan the Lonely Shepherd is standing under a special **tree** with N nodes, out of which K are blue and the rest are red. Bogdan really wants a unicorn, being tired of so many sheep and he thinks that if he could do some magic with this tree, a unicorn will show up. What kind of magic, you ask? He has to toggle a set of nodes from blue to red or red to blue (doing this twice cancels any change) and then cut an edge of the tree, resulting in two trees. Bogdan thinks that a pair of trees is unicornish (magic) if the trees have the same number of blue nodes. You need to output a set of toggle operations, and the edge to be cut, such that at the end the resulting two trees have the same number of blue nodes.

AHA! You thought this was gonna be an easy task, but there's a catch! Bogdan is colorblind! So he doesn't know the initial coloring of the tree, except for the fact that there are K blue nodes. So he asks you to find a set of toggle operations and a cut such that for any possible initial colouring of the tree, the resulting pair of two trees is unicornish! In case this is not possible, you must tell him so, because pointless hope is a dangerous thing.

Standard input

The first line will contain two integers, N and K.

The following $N - 1$ lines will each contain two numbers x and y meaning there's an edge between nodes x and y in our tree.

Standard output

In case there is no way of obtaining the unicornish pair, output `-1`, otherwise, output at most 10^6 lines, each line denoting a toggle operation or the final cut.

A toggle operation will be represented by a line starting with the string `flip` followed by a number x, denoting the node to be toggled, both separated by a space.

The cut operation will be represented by a line starting with the string `cut` followed by two numbers x and y denoting the endpoints of the edge to be cut, all separated by spaces. There can be at most one cut operation in the output.

Constraints and notes

- $2 \leq N \leq 10^5$
- $1 \leq K < N$
- Big thanks to the actual Bogdan Ciobanu for being a true superman in the making of this contest

Input	Output	Explanation

`3 1`
`1 2`
`2 3`
	`flip 1`
	`cut 2 1`
		For every valid coloring of the tree (with only one blue node), changing the color of the first and then cutting the $(1, 2)$ edge will leave us with two trees with the same number of blue nodes

`5 2`
`1 2`
`1 3`
`1 4`
`1 5`
	`-1`	There is no way to apply the operations such that for any initial valid coloring (with only two blue nodes), in the end we'll end up with two trees with the same number of blue nodes.

Luck-i-flip

Time limit: *1280 ms*
Memory limit: *264 MB*

In Sierra island there is a famous game called **Luck-i-flip**. To place a bet, each participant fills a row of length n with `0` s and `1` s, representing the two sides of a coin. The bets are public to prevent multiple participants from placing the same bet.

On the day of the event, the president of the Sierra Cultural Association flips a coin n times and the winner is the participant with the most correct coin side guesses. In case of a tie, the association keeps the jackpot.

Sara does not like to gamble but she is interested in the fact that this game is not solely reliant on luck. Not every bet offers the same chances of winning. In particular, she wants to know what are the odds of winning for the worst bet on the public ledger.

For that, Sara should count how many different draws will give the prize to the worst bet. Can you help her?

Take, for example, the three bets `001`, `110` and `101`:

- for the bet `001` the winning draws are `000`, `001` and `011`
- for the bet `110` the winning draws are `010` and `110`
- for the bet `101` the only winning draw is `101`

In this case, the worst bet is `101` and the number of different winning draws for the worst bet is 1.

Standard input

The first line is the number of bets, b in the public ledger, followed by a bet in each line.

Standard output

The output is the minimum number, m, of winning draws for the worst bet.

Constraints and notes

- $n \leq 22$
- $b \leq 100$
- $m \leq 20$

Input	Output	Explanation
2 00 11	1	There are two bets, `00` and `11`. There are four possible draws of length 2: `00`, `01`, `10`, `11`. If the draw is `00`, then the winning bet would be `00`. If the draw is `11` the winning bet would be `11`. But if the draw is `01` or `10` then both bets failed one coin side and lose the jackpot. For each bet, there is only 1 draw leading it to the prize. So the answer is the minimum which is 1
2 00 01	2	There are two bets, `00` and `01`. There are four possible draws of length 2: `00`, `01`, `10`, `11`. Both draws `00` and `10` would be won by `00`. Conversely, both draws `11` and `01` would be won by `01`. So both bets have 2 winning draws. The answer is the minimum which is 2

Googolplex

Time limit: *440 ms*
Memory limit: *264 MB*

The Alpha universe is much larger than our current universe and the distance between different planets is in the order of a few googolplex meters. A googolplex is a large number equal to $10^{10^{100}}$. In decimal notation, it is written as the digit 1 followed by 10^{100} zeros. Light travels at a speed of 3×10^8 meters per second. Hence, even light would take a really long time to travel one googolplex meters! Humans in the Alpha universe have the technology to travel at the speed of light, but since the distance between planets is still very large. The time to travel between different planets is more than the life span of normal humans, making it impossible to travel between planets directly.

Fortunately, in the Alpha universe there are some wormholes which shorten the distances by acting as tunnels connecting Earth to different planets. To travel from Earth to a distant planet which is X googolplex meters away, you would need to take the wormhole Y to shorten the distance. In this universe, the time t, in seconds, spent in the wormhole is given by the formula $t = (X^{\text{googolplex}+T})$ mod 10^Y where T is the time of the day (the time in seconds after 12:00 UTC).

For example, if a planet is 5 googolplex meters away and is using wormhole 2, then the time spent in wormhole is given by $t = (5^{\text{googolplex}+T})$ mod 10^2 seconds at the time of day T. Output the minimum time spent in the wormhole by determining the optimal time of day T.

For example, at time of the day $T = 10$, the time spent in the wormhole is $(5^{googolplex+10})$ mod 10^2. T can vary from 0 to 86399.

Standard input

The first line contains an integer N, the number of test cases.

Each of the next N lines contains two integers X and Y.

Standard output

Output a single integer for each test case denoting the minimum time spent in the wormhole.

Constraints and notes

- $1 \leq N \leq 20$
- $1 \leq X \leq 10^8$
- $1 \leq Y \leq 9$
- $0 \leq T \leq 86399$

Input	Output	Explanation
1 2 4	16	There is 1 test case. The distant planet is 2 googolplex meters away and the wormhole used to travel is 4. Hence, the distance is $(2^{googolplex+T})$ mod 10^4. The minimum time is 16 seconds when time of day $T = 4$.

3
5 1
100 3
3 5
	5
	0
	1
		There are 3 test cases. The expected outputs are `5`, `0`, and `1`.

Sic Semper Tyrannosaurus!

Time limit: *1280 ms*
Memory limit: *264 MB*

After finding out that a senator that cannot chew and senator W have conspired against Herr. Tyrannosaurus, Caesar had to ditch his efforts to impress Tiranca and come to give them an outright Promethean punishment! He gave them a sequence v of N numbers, an initial number K and Q updates and queries of the following form:

- ***Update***: Change the value of an element
- ***Query***: You are given two numbers l and r. A *tambourinishment* is the operation of taking a subarray $v_l, v_{l+1}, \ldots, v_r$ and transforming each of its elements $v_{l \le i \le r}$ into $\sum_{k=l}^{i} v_k$ (in some more elevated circles, they call this applying partial sums only to the elements of the given subarray). You have to output the value of v_r after tambourinishing the subarray K times. As these values can get quite large, you have to output their remainder after being divided by $10^9 + 7$. The query is not persistent (after applying the tambourinishments, the sequence goes back to its state before the query).

Standard input

The first line will contain the numbers N, K and Q in this order. The second line will contain N numbers representing the values of sequence v The i-th of the next Q lines will describe updates and queries in the following format: the line will begin with the character `Q` or `U`. In case the character is `Q`, the line will describe a query and the character will be followed by two numbers l and r mentioned in the problem statement. In case the character is `U`, the line will describe an update and the character will be followed by two numbers p and x implying that the value of v_p will be changed with x.

Standard output

The output will contain the answers of the queries, each written in order

Constraints and notes

- $1 \le N, Q \le 10^5$
- $2 \le K \le 8$
- The elements of the sequence will never exceed 10^9
- For all queries, $1 \le l \le r \le N$
- For all updates, $1 \le p \le N$
- Note: the author knows that the title of the problem is not fully adequate to the problem statement, but couldn't resist making that pun, also *tambourinishment* is a made-up word that the author had to use after being forced by his girlfriend.

Input	Output	Explanation

```
5 3 4
3 1 4 1 5
Q 1 5
Q 2 4
U 4 6
Q 3 5
```
	```
	87
	19
	47
	```
		We have 3 queries and 1 update.
		
		The last query is performed on the following subarray: $[4, 6, 5]$.
		
		After each of the 3 steps, the subarray looks as the following:
		
		$k = 1 : [4, 10, 15]$
		
		$k = 2 : [4, 14, 29]$
		
		$k = 3 : [4, 18, 47]$
		
		The result is 47.

Ave Caesar!

Time limit: *1280 ms*
Memory limit: *264 MB*

Tired of making ciphers and brutalizing triangles, Caesar decided to impress Tiranca the unicornosaurus with something else, his obsession with algebraic closures! Skeptic of his skill, Tiranca gives him the following problem:

A *valid string* is either:

1. A string consisting of a single character
2. The concatenation of 2 *valid strings* A and B, if $A \leq B$ lexicographically.

Given N strings, decide which are *valid strings*.

Standard input

- The first line contains an integer N, the number of given strings.
- The following N lines contain a string in each line.

Standard output

The output should contain a single line with N characters, the i-th being `1` if the i^{th} string from the input is valid, `0` if not.

Constraints and notes

- The total length of the strings will be smaller than 10^6
- All strings contain only uppercase characters

Input	Output
5 BBBABB AABAAB AABAA BBABAB ABBBBA	01000

Protecting the Hive

Time limit: *680 ms*
Memory limit: *264 MB*

The Alfa rabbit has a bee friend that provides him with honey, called Maya. Maya is the queen of the bees and she belongs to the hive. The hive is the place where the worker bees produce and conserve honey.

The hive is made of cells, each cell has the form of a hexagon where each side of this hexagon measures 1 cm. The hive has N rows, where every odd row has M cells and every even row has $M - 1$ cells. Each cell can be active or inactive. An active cell is where the bees can produce honey, while an inactive cell it has some problem and the bees cannot produce honey there.

The hive is composed of many sections, where a section is made of a set of connected cells (two cells are connected if they are adjacent to each other through one of their 6 sides). Maya is able to:

1. Create a perimeter with a special substance for each section in order to protect it from other insects.
2. Activate an inactive cell.

From time to time, Maya wants to know, given a specific cell, what is the perimeter of the section it belongs to. Can you help the queen?

Standard input

The input will consist of the numbers N and M, where N is the number of rows and M is the number of columns for the odd rows, followed by N rows of numbers.

Row 1 consists of M numbers (either 0 or 1, where 0 represents an inactive cell and 1 represents an active cell), row 2 consists $M - 1$ numbers and so on.

On the next line we have the number Q that represents the number of queries, followed by Q query lines, where each query is of the form " `a` row column" or " `k` row column". If the first letter of the query is `a` it means that the queen wants to activate the cell $(\text{row}, \text{column})$, otherwise, if the first letter of the query is `k` the queen wants to know the perimeter of the section that the cell $(\text{row}, \text{column})$ belongs to.

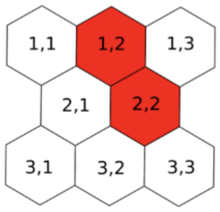
Standard output

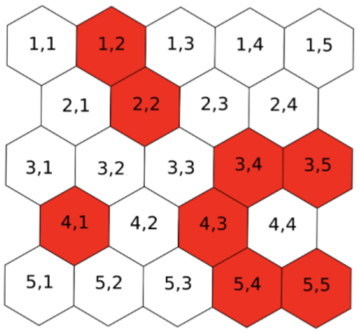
The output consists of Q lines where every line represents the answer to each query where the queen wants to know the perimeter of the section.

Constraints and notes

- $1 \leq N \leq 100$
- $1 \leq M \leq 100$
- $1 \leq Q \leq 10^5$
- The queries will consist of only valid cells.

Input	Output	Explanation

```
3 3
0 1 0
0 1
0 0 0
5
a 1 3
k 2 2
k 2 1
a 3 1
k 3 1
```
	```
	12
	0
	6
	```
		

```
5 5
0 1 0 0 0
0 1 0 0
0 0 0 1 1
1 0 1 0
0 0 0 1 1
8
a 2 1
k 2 1
k 1 2
k 3 4
k 4 1
k 5 5
a 3 3
k 3 3
```
	```
	12
	12
	22
	6
	22
	34
	```
		 1,1
		1,2
		1,3
		1,4
		1,5
		2,1
		2,2
		2,3
		2,4
		3,1
		3,2
		3,3
		3,4
		3,5
		4,1
		4,2
		4,3
		4,4
		5,1
		5,2
		5,3
		5,4
		5,5

Bearcity Renting

Time limit: *1280 ms*
Memory limit: *264 MB*

Vangelis the Bear and its friend Charlie want to open their own store in Bearcity. They need to decide which place they will rent to start their store. Since they will constantly need to buy supplies to the store, they want to find a place near a warehouse.

Bearcity used to be a small town, so all the streets are only one way and there is always a heavy traffic jam. The new mayor, Lisa the bear, seeing that the city is expanding, wants to turn some one-way streets into two-way streets. Some conversion cost is associated with each street. Lisa wants to make some conversions in a way that the total cost of conversion is minimized, keeping in mind that all warehouse points should be connected by two-way roads.

Vangelis and Charlie, after coming to know about this idea, decided to rent a place on a two-way street, so clients can come and go easily from the store. As there can be more than one conversion plan, help Vangelis and Charlie decide how many streets are part of all possible conversion plans.

Standard input

The first line of the input contains two positive integers. The first one indicates the number N of warehouses considered in the conversion plan and the second one the number M of streets. The next M lines are triplets indicating, for each street, its endpoints u and v, followed by its conversion cost.

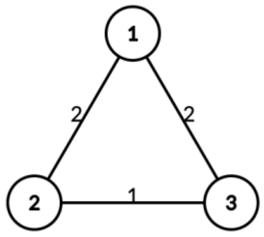
Standard output

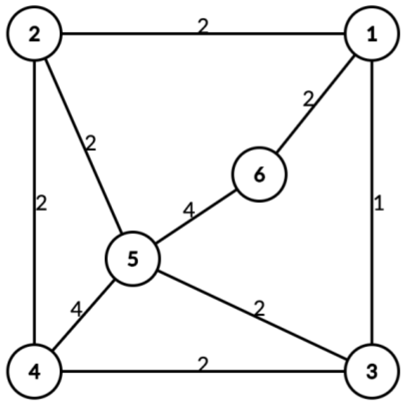
Output a single value representing the number of streets that are part of all possible conversion plans.

Constraints and notes

- $1 \le N \le 10^5$
- $1 \le M \le 2 * 10^5$
- The cost of conversion c_s for each street s: $1 \le c_s \le 10^4$
- There is at most one road connecting two warehouses
- It is guaranteed at least one conversion plan exists

Input	Output	Explanation

```
3 3
1 2 2
3 1 2
2 3 1
```
	1	 There are 2 possible minimum cost conversion plans: $(1,3),(2,3)$ or $(1,2),(2,3)$. Only the street $(2,3)$ occurs in both of them.

```
6 10
1 2 2
1 3 1
1 6 2
2 4 2
2 5 2
4 3 2
3 5 2
4 5 4
5 6 4
6 6 2
```
	2	 Only the roads $(1,3)$ and $(1,6)$ are part of all possible conversion plans.

Yin Yang

Time limit: *1280 ms*
Memory limit: *264 MB*

Finding the perfect balance is something sought after by many people and in many ways... sometimes even in strings. We'll call a string unbalanced if it has even length and its two halves are equal. Find a string of length N, consisting only of characters `y` and `Y`, such that it has as few distinct unbalanced substrings as possible.

Your score per test will be computed as $(1 + \frac{1}{10})^{-K}$, where K is e^{F-O}, O is the optimal number of distinct unbalanced substrings and F is the number of distinct unbalanced substrings you have obtained.

Perfectly balanced as all things should be.

Standard input

The first line contains an integer N.

Standard output

Print the answer on the first line.

Constraints and notes

- $1 \leq N \leq 300$
- By e we mean Euler's number, which is ≈ 2.718282

Input	Output
4	yyyY

Infinite String

Time limit: *1280 ms*
Memory limit: *264 MB*

Ivan Likes playing with strings. Today he decided to make the biggest string ever.

Using the first B characters of the Latin alphabet, he wants to write a string that first consists all the words of length 1, in lexicographical order, then all the words of length 2, in lexicographical order, etc.

For example, using the first 4 characters (a, b, c, d), the string will be the following (without spaces):

a b c d aa ab ac ad ba bb bc bd ca cb cc cd da db dc dd aaa aab aac aad aba ...

Because the string is infinite and Ivan has to go to the University in the morning, he is interested what character is at index X.

Standard input

The first line contains one integer, T, the number of queries.

The next T lines contain two integers, B and X, that correspond to the number of characters used in the alphabet and the 0-based index of the character Ivan wants to know.

Standard output

Output T lines. The ith line has the character for the ith query.

Constraints and notes

- $1 \leq T \leq 100$
- $1 \leq B \leq 26$
- $0 \leq X \leq 10^{18}$

Input

```
6
4 5
4 3
2 32
26 24
26 50
4 27
```

Output

```
a
d
b
y
a
d
```

Explanation

For cases where $B = 2$, the 32th character is the following:

abaaabbabbaaaaababaabbbaababbbab **b** baaaaaaabaabaaabbabaaabab ...

For Cases where $B = 4$, the 3rd, 5th and 27 th characters are the following:

abc **d** a **a** abacadbabbbcbdcacbccc **d** dadbdcddaaaaabaacaadaba ...

For cases where $B = 26$, the 24th and 50th characters are the following:

abcdefghijklmnopqrstuvwx **y** zaaabacadaeafagahaiajakal **a** manaoapaqar ...

Note that all indices are 0-based.

Concentration Game

Time limit: *1280 ms*
Memory limit: *264 MB*

Concentration is a card game in which a set of cards are laid face down on a surface and two cards are flipped face up over each turn. The objective of the game is to turn over pairs of matching cards.

There will be $2N$ cards of N different types labelled with integers from 1 to N.

Interaction

You should read an integer N.

A move consists in choosing two distinct card indices i_1 and i_2 and revealing them. To make a move, print a line with both indices separated by a space between them (and don't forget to flush the output!). If these cards have the same value written on them, the interactor will reply with `MATCH` and eliminate the cards from the surface. Otherwise, it will reply with two integers, the first representing the value of i_1 and the second representing the value of i_2.

If at any point in time the interactor decides you've made an invalid move, it will print `-1` and exit. You must read this value and exit right away, otherwise you might get a verdict which does not properly reflect your invalid move.

You must print `-1` after you have decided to not make any more moves.

Constraints and notes

- $1 \leq N \leq 10^3$
- You are allowed at most $2N$ moves
- You may not ask to make a move with a card that has already been removed from the surface
- The card indices **DO NOT** change after other cards are removed from the surface

Interaction		Explanation
`5`		The initial order for the cards is $[3, 4, 3, 4, 1, 2, 5, 5, 1, 2]$.
	`1 2`	
`3 4`		
	`2 4`	
`MATCH`		
	`1 3`	
`MATCH`		
	`9 10`	
`1 2`		
	`5 9`	
`MATCH`		
	`8 7`	
`MATCH`		
	`10 6`	
`MATCH`		
	`-1`	

Monokeros

Time limit: *3080 ms*
Memory limit: *264 MB*

The true tyrant, mr. W has given Tiranca a new problem: you are given an initially empty binary search tree and a sequence of numbers $(x_1, x_2, ..., x_N)$. A binary search tree is a binary tree, that stores a value in each node and respects the following rules:

- the value stored in its left child is smaller or equal to the value of the node
- the value stored in its right child is strictly greater than the value of the node

Your task is to insert these numbers in the binary search tree (in the given order) and output the depth (edge-distance from the node to the root) of the newly added node after each insertion. An insertion goes like this:

```
insert_value(current_node, new_value):
    if new_value <= value(current_node):
        if the left child of current_node exists:
            insert_value(left_child(current_node), new_value)
        else:
            // create a new node with the new_value and place it as current_node's left child
    else:
        if the right child of current_node exists:
            insert_value(right_child(current_node), new_value)
        else:
            // create a new node with the new_value and place it as current_node's right child

```

Standard input

The first line will contain N, the number of elements in the sequence.

The second line will contain N numbers: $x_1, x_2, ..., x_N$.

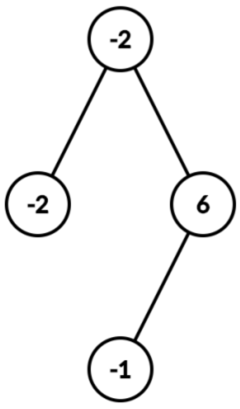
Standard output

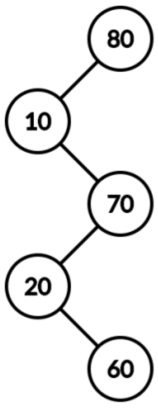
The first line will contain N numbers, corresponding to the depths of the added nodes.

Constraints and notes

- $1 \le N \le 2 \cdot 10^5$
- $-10^9 \le x_i \le 10^9$

Input	Output	Explanation

`4`
`-2 -2 6 -1`
	`1 2 2 3`	

`5`
`80 10 70 20 60`
	`1 2 3 4 5`	

Batchman and GCD

Time limit: *1280 ms*
Memory limit: *264 MB*

The streets of Gotham have been poisoned by crime and implementation problems again! But rest assured, our hero, Batchman has come to save the day! Today, he has to face the terrible XML parsing monster by countering him with a number theory problem. The issue is that while fighting the terrible creature, he can't think of an efficient solution for the problem, so he asks for your help!

The problem goes like this: you are given an array $A = (a_1, a_2, ..., a_N)$ and a number K. Consider all subsequences of A of size at most K. For each subsequence, compute the GCD of the numbers in the subset. How many **distinct** GCDs can you get?

Standard input

The first line will contain the numbers N and K.

The second line will contain N integers corresponding to the elements of A.

Standard output

The output will contain a single number corresponding to the problem's answer.

Constraints and notes

- $1 \le N \le 10^5$
- $1 \le K \le N$
- $1 \le a_i \le 10^5$

Input	Output	Explanation
5 5 4 8 12 16 18	7	The only GCD values that can be obtained are: 2 = gcd $\{4, 18\}$, 4 = gcd $\{4\}$, 6 = gcd $\{12, 18\}$, 8 = gcd $\{8, 16\}$, 12 = gcd $\{12\}$, 16 = gcd $\{16\}$, 18 = gcd $\{18\}$

Impact Factor

Time limit: *1280 ms*
Memory limit: *264 MB*

IEEE Xplore

Warm greetings to all IEEExtreme Participants from the Xplore API Team!

As part of the IEEExtreme Competition, you will be tasked with a programming challenge to determine the Impact Factor of an academic journal. The input content file of this exercise is static (non changing) to allow for consistent results during this competition.

For a full dynamic database search IEEE Xplore API is available for your IEEE research needs. Xplore API provides metadata on 4.9mm academic works and is now delivering full-text content on 50k 'Open Access' articles. Xplore API will make your research needs fast and easy. The Xplore API Portal supports PHP, Python and Java as well as providing output in Json and XML formats. Many API use cases are listed within the API Portal.

Xplore API registration is free. To learn more about IEEE Xplore API please visit developer.ieee.org/ and register for an API key TODAY!

Challenge

The impact factor (IF) of an academic journal is a scientometric index which reflects the yearly average number of citations to recent articles published in that journal. In any given year, the impact factor of a journal is the number of citations, received in that year, of articles published in that journal during the two preceding years, divided by the total number of **citable items** published in that journal during the two preceding years:

$$\mathrm{IF}_y = \frac{\mathrm{Citations}_{y-1} + \mathrm{Citations}_{y-2}}{\mathrm{Publications}_{y-1} + \mathrm{Publications}_{y-2}}$$

Standard input

The first line is the number of records to process. The second line is the publications to be included in the report. The **publicationNumber** is the unique key to identify the publication. The **publicationTitle** is the title of the publication. The **articleCounts** array contains the **year** and **articleCount** for that year and publication.

The subsequent lines are JSON packets containing citations for random publications. The **paperCitations** contains an array, called **ieee**, with information about the cited papers. The **publicationNumber** in the array is the publication number of the publication the cited article appears in. The **year** is the year the article was published.

Standard output

List the publication title and its Impact Factor. Order by descending Impact Factor. Format the output with one journal name per line and with the name and the score (with two decimal places) separated by a colon. Example: `Letters on IEEEXtreme: 1.78`

In case multiple publications have the same impact factor, print them in alphanumerical order.

Constraints and notes

- The maximum number of records is 1001

Input	Output	Explanation
`6`	`Letters on IEEEXtreme: 1.78`	Taking the second line of the input you must generate the Impact Factor for publications **Letters on IEEEXtreme** and **Journal of 24 hours programing**. **Letters on IEEEXtreme** had the greatest Impact factor so it appears first in the output.
`{"publications": [{"publicationTitle`	`Journal of 24 hours programing: 1.20`	
`{"publisher": "IEEE","title": "Publi`		
`{"publisher": "IEEE","title": "Publi`		
`{"publisher": "IEEE","title": "Publi`		
`{"publisher": "IEEE","title": "Publi`		
`{"publisher": "IEEE","title": "Publi`		

Alfa Universe Weight Sorting

Time limit: *1280 ms*

Memory limit: *264 MB*

There are several races in Alpha universe and each one of them has a different weight. For some pairs of races, you are given the result of comparing their weight. From transitivity, you can deduce the order of some of the pairs that you are not given, but other pairs are not known. For example, if $A > B$ and $B > C$, you can deduce that $A > C$.

A *valid ordering* is a permutation of all the races, such that for each pair (A, B), if $A < B$ then A's index in the permutation is smaller than B's index.

Someone is asking you to compare the weight of two races that cannot be deduced from the known compared pairs. You want to answer it by providing a minimal number of *valid orderings*, such that:

- For every pair (A, B), where it cannot be deduced which of them is lighter and which is heavier, there is at least one ordering where A comes before B, and at least one ordering where B comes before A.

For example, if there are four races: A, B, C and D and you know that:

1. The weight of C is less than that of B;
2. The weight of B is less than D;
3. And that the weight of C is less than A.

In this case the six pairs are divided into the four known orders:

1. $C < A$
2. $B < D$
3. $C < B$
4. $C < D$

Two are unknown:

1. $A?B$
2. $A?D$

You can provide the following two *valid orderings*:

- $C < A < B < D$
- $C < B < D < A$

Where all these permutations obey all four known weight order relations and for any unknown pair there exists at least one permutation that cover each order of the pair.

Your task is to find the minimal number or permutations, N, and print the N *valid orderings*.

Standard input

The first line contains the number of races, N, and the number of known comparisons M.

The following M lines contains the pairs in the format XY where X and Y are uppercase letters representing the race.

Standard output

The first line is the minimal number K of permutations needed.

Each of the next K lines should contains a permutation of the first N letters of the alphabet.

- $K = 0$ means that the problem can not be solved (the known comparisons are not transitive).
- $K = 1$ means that there are no undetermined pairs and the next line is the only permutation that adheres to all the given constrains.

Constraints and notes

- $1 \leq N \leq 26$ The races are denoted by the first N letters in the English alphabet.
- $0 \leq M \leq 325$ You may assume that the input is in the correct format
- **This is an NP problem and it may not be possible for anyone to solve all the test cases within the time limits**

Input	Output	Explanation
4 4 CA BD CB CD	2 CABD CBDA	This sample is the example given above.
3 2 CA AB	1 CAB	The given comparisons allow for only one permutation: $C < A < B$
10 10 AB AJ DF DC FA AC EB IH JD AI	0	Take the 3rd, 5th, 2nd and 10th comparisons and we get a contradiction: $D < F < A < J < D$

Xranda and Tree

Time limit: *1280 ms*
Memory limit: *264 MB*

Xranda and a man who does not chew are living in an tree with N nodes, with numerical labels on the edges. Their passion for treeology determined them to answer some seemingly impossible problems like "how many labeled trees lie in the isomorphism class of a given tree?" or "what is the generalised rotation distance from their tree and some arbitrary other one?". Exhausted by these tasks, they ask you to solve an easier one.

We define the distance $d(u, v)$ between two nodes u and v as the largest label of an edge which belongs to the unique path between them. Compute the sum of the distances across each possible pair of nodes, that is:

$\sum_{a=1}^{n} \sum_{b=1}^{a-1} d(a, b)$

Given that this number can be quite large, we are only interested of its remainder when divided by $10^9 + 7$.

Standard input

The first line will contain the number of nodes of the tree, N.

The next $N - 1$ lines will contain the description of the edges of the tree, that is, the line $i + 1$ will contain (in order) numbers A_i, B_i and W_i, meaning that there is an edge between nodes A_i and B_i with the label W_i between them.

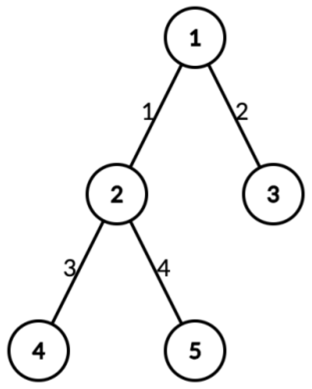
Standard output

The output will only contain the desired sum.

Constraints and notes

- $2 \leq N \leq 10^5$
- $1 \leq W_i \leq 10^9$

Input	Output	Explanation

```
5
1 2 1
1 3 2
2 4 3
2 5 4
```
	<pre>30</pre>	 We have the following distances: $d(1, 2) = 1$; $d(1, 3) = 2$; $d(1, 4) = 3$; $d(1, 5) = 4$; $d(2, 3) = 2$; $d(2, 4) = 3$; $d(2, 5) = 4$; $d(3, 4) = 3$; $d(3, 5) = 4$; $d(4, 5) = 4$ The total sum is 30.

Reshuffle Teams

Time limit: *1280 ms*
Memory limit: *264 MB*

Recently Sara invented a new board game, and she has invited all her friends to play at her house. Everyone is already sitting at a big circular table. Sara randomly attributes a team, represented by a letter from `A` to `D`, to each friend. In order to group teams around the table, a subset of players should get up and change their places.

What is the minimum number of people that need to change their seat such that all the players in any team occupy a contiguous sequence of chairs?

Standard input

The first line of the input contains the number of times, T, that you need to solve the challenge.

Each of the following T lines contains a string S_i of length L_i.

Standard output

For each test, output a line containing a single integer representing the minimum number of people that have to change their seat.

Constraints and notes

- $1 \leq T \leq 10$
- $1 \leq L_i \leq 10^5$
- Teams are represented by characters (`A`, `B`, `C` or `D`)

Input	Output	Explanation

```
5
CBBACC
DBCA
CCACACC
ABCABCABC
ABCDABCDABCDABCD
```
	```
	0
	0
	2
	6
	12
	```
		CBBACC -> 0 (Teams are already sitting beside each other.)
		
		DBCA -> 0 (Teams are already sitting beside each other.)
		
		CCACACC -> CCCAACC: 2 letters out of place
		
		ABCABCABC -> AAABBBCCC: 6 letters out of place
		
		ABCDABCDABCDABCD -> AAAABBBBCCCCDDDD: 12 letters out of

Pair Swap Teams

Time limit: *1280 ms*
Memory limit: *264 MB*

In a parallel universe, Sara also invented a new board game, and she has invited all her friends to play at her house. Everyone is already sitting at a big circular table. Sara randomly attributes a team, represented by a letter from `A` to `D`, to each friend. In order to group teams around the table, a subset of players should get up and change their places.

Contrary to the other universe, they have to swap places in pairs.

What is the minimum number of swaps such that all the players in any team occupy a contiguous sequence of chairs?

Standard input

The first line of the input contains the number of times, T, that you need to solve the challenge.

Each of the following T lines contains a string S_i of length L_i.

Standard output

For each test, output a line containing a single integer representing the minimum number of seat swaps.

Constraints and notes

- $1 \leq T \leq 10$
- $1 \leq L_i \leq 10^5$
- Teams are represented by characters (`A` , `B` , `C` or `D`)
- The same player can be involved in multiple seat swaps

Input	Output

```
5
CBBACC
DBCA
CCACACC
ABCABCABC
ABCDABCDABCDABCD
```
	```
	0
	0
	1
	3
	6
	```

Unicornosaurus

Time limit: *1040 ms*
Memory limit: *264 MB*

Tiranca the unicornosaurus broke Sherbot's duckinator*. After close inspection, she realized that there were N intervals $[L_i, R_i]$ of broken planks. In each interval, all of the planks numbered from L_i to R_i were completely destroyed.

Luckily, Tiranca is a unicornosaurus and she possesses M superpower actions to repair the duckinator. Each superpower costs C_i rainbows and can be used to repair all the planks in the interval $[A_i, B_i]$. To fix the duckinator, she has to fix all of the broken planks and, because rainbows are expensive, she should use the smallest possible number of rainbows.

1* *Duckinator*: Long fence made of wide planks used to catch ducks. The planks are arranged in a line, numbered from 1 to S.

Standard input

The first line contains three integers, N, M and S. The i-th following N lines contain two integers, L_i and R_i, describing the damage done to the duckinator. The i-th following M lines contain three integers A_i, B_i and C_i, describing Tiranca's possible superpower actions.

Standard output

The output should contain the minimum number of rainbows required to repair the duckinator. In case it is impossible to fix, the output should be `-1`.

Constraints and notes

- $1 \le S \le 10^5$
- $1 \le C_i \le 10^9$
- $1 \le L_i \le R_i \le S$
- $1 \le N, M \le 10^5$
- The intervals of broken planks **may overlap**.
- The author made up the term **duckinator**.

Input	Output	Explanation

```
1 3 15
5 10
3 7 2
6 12 5
2 11 6
```
	`6`	There is one broken segment: $[5, 10]$.
		We choose $[2, 11]$ with the total cost of 6 to fix it.
		Note that another solution would be choosing $[3, 7]$ and $[6, 12]$, but the total cost of that would be 7.

```
2 4 15
3 7
9 10
2 6 10
3 9 15
5 12 13
8 10 30
```
	`23`	There are two broken segments: $[3, 7]$ and $[9, 10]$.
		We choose $[2, 6]$ and $[5, 12]$ with the total cost of 23 to fix it.

Strictly Convex Pairs

Time limit: *3680 ms*
Memory limit: *264 MB*

There is a polygon with N vertices in standard 2-D Cartesian coordinates. The polygon is **strictly convex**. That is, the polygon is convex, and there are no three collinear vertices (i.e. lying in the same straight line).

There are M other distinct points which are located strictly inside or outside the polygon (not on polygon edges). Andy wants to pick two different points a and b from those M points so that he can connect a and b with a straight **line segment** without intersecting or touching the polygon. Andy is curious about how many different unordered pairs of points a and b he can pick. Can you help Andy to count that?

Note: An unordered pair (a, b) is the same as the pair (b, a).

Standard input

The first line of the input contains two integers, N and M.

The next N lines describe the convex polygon. Each line contains two integers indicating x-coordinate and y-coordinate of one vertex. The vertices are given in counter-clockwise order.

The next M lines describe points that Andy can pick. Each line contains two integers indicating x-coordinate and y-coordinate of one point.

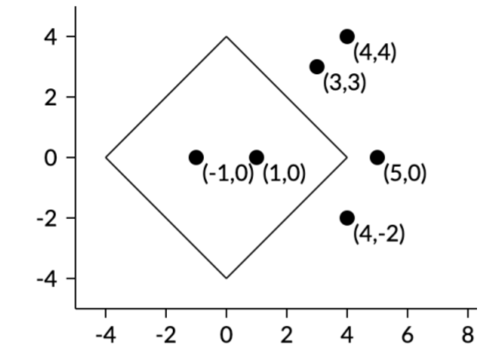
Standard output

Print one integer representing the number of pairs.

Constraints and notes

- $3 \leq N \leq 2 \times 10^5$
- $2 \leq M \leq 2 \times 10^5$
- $-10^9 \leq x$-coordinate, y-coordinate $\leq 10^9$
- The polygon is strictly convex.
- All M points are distinct and not located on any polygon edges.

Input	Output	Explanation

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

The five pairs that can be picked are:

- point $(5, 0)$ and point $(3, 3)$
- point $(5, 0)$ and point $(4, -2)$
- point $(5, 0)$ and point $(4, 4)$
- point $(1, 0)$ and point $(-1, 0)$
- point $(3, 3)$ and point $(4, 4)$

Charlie New PC

Time limit: *2480 ms*

Charlie has been given money to build a new PC. He needs your help to select components, using as much of the money as possible, without going over the budget. There are a number of component types to choose from (e.g. RAM, hard drive, processor, etc.), and he must select exactly one of each component type for the computer.

Standard input

The first line of the input specifies an integer T, the number of test cases. Each test case begins with an integer B, which represents the maximum amount that Charlie can spend. The next line contains an integer N, which represents the number of component types. The next line contains N space separated integers that represent the number of options K_i, for each component type $1 \leq i \leq N$. Each of the next N lines contain the costs for the components. The i^{th} line contains a space separated list of K_i integers indicating the costs of each component option.

Standard output

For each test case, you should output a single line with the maximum cost of the computer, using exactly one of each component types. The cost needs to be less or equal to the maximum amount available B. If there is no configuration that can be constructed for the given budget, the output should be `0`.

Constraints and notes

- $1 \leq T \leq 10$
- $1 \leq B \leq 2 * 10^9$
- $1 \leq N \leq 10$
- $1 \leq K_i \leq 100$
- The sum of all options $K_i \leq 100$
- The costs of the components are integers between 1 and $2 * 10^9$

Input	Output	Explanation

```
1
50
4
3 2 1 2
15 10 49
11 17
10
13 23
```
	`50`	In the sample input, there is one test case.
		
		This test case has a **b** budget of 50, and there are 4 types of components.
		
		• Three options for component 1, with cost 15, 10, and 49;
		• Two options for component 2, with cost 11 and 17;
		• One options for component 3, with costs 10;
		• Two options for component 4, with cost 13 and 23.
		
		The most expensive computer that Charlie can build will cost 50, choosing the 10 option for component 1, the 17 option for component 2, the 10 option for component 3, and the 13 option for component 4.

Puzzle of Rooks

Time limit: *1280 ms*

Memory limit: *264 MB*

Some predict that by the year 2050 AI will be able to solve most of the challenges we face today. You are a fan of the prediction, but your friend Lenny is not yet convinced. You therefore have to prove to her the power of AI by solving the *Puzzle of Rooks*!

There is a chessboard of infinite size. Each cell is identified based on its horizontal and vertical coordinates using an integer pair (x, y). There are n rooks located at n initial cells on the chessboard. There are additionally n target cells. To solve the puzzle, one must move the n rooks, so that each of the target cells contains exactly one rook, and no two rooks attack each other at any time. Two rooks attack each other if they share a same row or column, i.e. their cells have a same x or a same y coordinate. The following operations may be performed:

- Move a rook on the chessboard to one of its horizontally or vertically adjacent cell. That is, moving a rook at cell (x, y) left, right, up, or down changes its cell to $(x - 1, y), (x + 1, y), (x, y + 1), (x, y - 1)$ respectively.
- Record the position of a rook and temporarily remove it from the chessboard. The cell of the rook becomes empty. At most one rook can be temporarily removed at a time. In other words, the chessboard always has at least $n - 1$ rooks on it.
- Put the removed rook back to its recorded position on the chessboard. A new rook can be removed after a previously removed rook is put back.

You are to create an AI that will solve the Puzzle of Rooks. The AI shall find a sequence of operations that will move the n rooks to the n target positions without letting the rooks attack each other during the whole process. Lenny is also very picky. She will only be convinced if your AI is efficient enough. More specifically, your AI shall solve the puzzle with no more than 2050 operations!

Standard input

The input has one integer n on the first line.

Each of the next n lines has two integers describing the initial cell of a rook.

Each of the next n lines has two integers describing a target cell.

Standard output

Output a single integer k ($k \leq 2050$) on the first line, the number of operations your AI needs. Then print k lines, each line is in the format `x y op`. `op` is a single character describing the type of operation performed on cell (x, y):

- `L`, `R`, `U`, `D`: Move the rook currently at cell (x, y) left, right, up, or down.
- `T`: Temporarily remove the rook at cell (x, y).
- `P`: Put the temporarily removed rook back to its recorded position (x, y).

Note that as the chessboard is infinitely large, it is allowed that a rook is moved to a cell with negative coordinates.

If the output sequence contains more than 2050 operations, or any of the operation provided is invalid (e.g. an operation results in two rooks attacking each other; attempting to move a rook at (x, y) but the cell (x, y) is empty; putting a removed rook back to a position different from what was recorded), your solution will receive *Wrong Answer*.

Constraints and notes

- $1 \leq n \leq 10$
- All coordinates of the initial and target cells are between 1 and 99 inclusive.
- No two initial cells share a row or column.
- No two target cells share a row or column.
- At least one target cell is not among the initial cells.
- It can be proved that for any initial and target cells satisfying the given constraints a sequence of no more than 2050 operations exists to solve the puzzle.

Input

```
2
1 1
2 2
1 2
2 1
```

Output

```
6
2 2 T
1 1 R
2 1 R
2 2 P
2 2 L
3 1 L
```

Input

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

Output

```
18
2 3 U
1 2 L
2 4 T
3 1 L
2 1 L
2 4 P
0 2 T
1 1 U
1 2 U
0 2 P
1 3 T
0 2 D
2 4 D
2 3 R
3 3 D
0 1 R
1 1 R
1 3 P
```

Minimum Permutation

Time limit: *1280 ms*

Memory limit: *264 MB*

You are given an array A of size N and a set S with M elements. Each number from 1 to $N + M$ occurs exactly once in either A or S.

You need to insert the elements from the set S into the array A in order to obtain the lexicographically smallest permutation.

Note: A Sequence $X_{1..n}$, of the same length as a sequence $Y_{1..n}$, is considered lexicographically smaller than $Y_{1..n}$ if, and only if, there exists an index j for which X_i = Y_i for $1 \leq i < j$, and $X_j < Y_j$.

Standard input

The first line contains the numbers N and M. The second line contains N integers corresponding to the elements of A. The third line contains M integers corresponding to the elements of S.

Standard output

The output should contain the elements of the lexicographically smallest permutation, separated by single spaces.

Constraints and notes

- $1 \leq N \leq 10^5$
- $1 \leq M \leq 10^5$
- $2 \leq N + M \leq 10^5$
- All the elements are unique and between 1 and $N + M$

Input	Output	Explanation

`3 2`
`3 1 5`
`4 2`
	`2 3 1 4 5`	The array is $[3, 1, 5]$ and the set is $\{4, 2\}$.
		The first operation is to insert 2 at the beginning of the array, obtaining $[2, 3, 1, 5]$.
		The second operation is to insert 4 right after 1, obtaining $[2, 3, 1, 4, 5]$.

Merchant Association

Time limit: *3680 ms*
Memory limit: *520 MB*

As the leader of the Xtreme Merchant Association you are trying to expand its already lucrative trading business. There are n towns in the state, numbered 1 to n. The towns are connected by $n-1$ bidirectional roads. All towns buy and sell the same type of goodies. Town i buys and sells one goodie for a price of p_i dollars.

Within the association, there are k merchants located at k different starting towns. Each merchant may choose a destination town and travel to it from his/her starting town. The merchant must visit each town on the chosen path exactly once and cannot go back. To avoid competition inside the Xtreme Association, no two merchants' travel paths can include the same town (in particular, one merchant cannot visit another merchant's starting town). A merchant may buy exactly one goodie at any town on the travel path, and sell that goodie at the destination town for a profit. Each merchant may only buy and sell once, but may choose not to travel and stay at the starting town if no profit can be made.

The goal of the association is for all the merchants together to make the largest possible profit. Look at the goodie prices closely and instruct the merchants about where their destinations should be and how they should buy and sell. What is the optimal total profit the merchants can make?

Standard input

The input has one integer n on the first line representing the number of towns.

The second line has n integers describing the prices at each town. The ith integer is p_i.

Each of the next $n-1$ lines has two integers x, y describing one bidirectional road connecting town x to town y.

Lastly there is a line with an integer k, the number of merchants. This is followed by a line with k distinct integers giving the starting towns of each merchant.

Standard output

Output a single integer, the largest total profit the merchants can make.

Constraints and notes

- $1 \leq n \leq 2 \times 10^5$
- $1 \leq p_i \leq 10^9$
- $1 \leq k \leq n$
- It is guaranteed that all towns are connected by roads.

Input	Output	Explanation

```
12
3 1 2 3 5 6 7 2 2 4 10 11
1 2
1 3
4 2
5 2
3 6
6 7
6 8
6 9
8 10
9 11
9 12
5
3 8 4 11 12
```
	`12`	• The merchant starting at town 4 travels to town 5, buying at town 2 for 1 dollar and selling at town 5 for 5 dollars (earning 4 dollars).
		• The merchant starting at town 3 travels to town 1, buying at town 3 for 2 dollars and selling at town 1 for 3 dollars (earning 1 dollar).
		• The merchant starting at town 8 travels to town 10, buying at town 8 for 2 dollars and selling at town 10 for 4 dollars (earning 2 dollars).
		• The merchant starting at town 11 travels to town 7, buying at town 9 for 2 dollars and selling at town 7 for 7 dollars (earning 5 dollars).
		• The merchant starting at town 12 does not travel.

Xtreme Rappers

Time limit: *200 ms*

Memory limit: *264 MB*

Rapper CrissyChris and her friend, JaCe, will participate in Gold Level, one of the most important events of freestyle in the world. They are training improvising 3-word phrases, where each one has to contribute at least one word.

They want to improvise the maximum number of phrases without repeating words (a word can be used only in one phrase), CrissyChris and JaCe select an assortment of words from Ctoke's database (Ctoke is one of the oldest rappers in freestyle and his database contains only unique words). CrissyChris and JaCe both selects K and J words from the database respectively. CrissyChris and JaCe never have words in common.

CrissyChris is an actual legend within IEEE and has supported the IEEEXtreme competition for the past 13 years in one way or another; whether it was all those late nights throughout the competitions or purchasing and shipping all of the top 100 prizes! She has recently moved on within IEEE to another department. This challenge is a tribute to her and all she has done to enrich the competition through the years! Farewell Christine Eldridge, you will be missed!

Standard input

One line with two integers K and J, representing the number of words available for each rapper.

Standard output

Print the maximum number of 3-word phrases they can improvise together.

Constraints and notes

- $0 \leq K, J \leq 10^{10}$

Input	Output	Explanation
`4 3`	`2`	With the sample input, we can form at most two 3-word phrases. One phrase can have 2 words from JaCe and 1 word from CrissyChris and the other phrase can have 2 words from CrissyChris and 1 from JaCe.
`0 10000000000`	`0`	Without words for CrissyChris no phrases can be created.

Lexical Addition

Time limit: *1280 ms*
Memory limit: *264 MB*

Michael is a kid who loves calculators. He is really obsessed with anything about digits and integers. He is also really curious about operations, such as addition, subtraction, division, and multiplication. Every day, he plays with a calculator, enters some numbers, performs some operations, and then the calculator gives the result. He repeats this pattern multiple times until he is bored.

One day, Michael discovers something interesting about the sum of integers. He notices that different pairs of numbers can be added together to give the same result. Pairs such as $5 + 7$ and $8 + 4$, which both are equal to 12. To further deepen this serendipitous discovery, he notices that this interesting property applies to groups of numbers greater than two, such as $4 + 6 + 6$ and $5 + 8 + 3$, which both are equal to 16. He wonders if such bizarre behavior can be observed to groups of four or five integers and, to his astoundment, he keeps increasing the size of the sequences and this property is always present!

He wonders if there is a way to reverse the whole process. He proposes the existence of an integer N, and guesses the sequence of integers that, when added, results in N. He found that for some integer N, his thought experiment could have multiple solution.

For example, 16 can be expressed from the sum of these sequences:

- $[1, 15]$
- $[1, 1, 14]$
- $[5, 5, 6]$
- $[4, 6, 6]$
- $[5, 8, 3]$
- and so on.....

Because the number of different configurations is so large, Michael wants to constraint the elements used in the sequences. He chooses another 2 integers A and B, so that every element of the sequence is an integer greater than or equal to A and also less than or equal to B. More formally, for every element X in the sequence, $A \leq X \leq B$. The element does not have to be unique.

Considering this constraint, if $N = 16, A = 4, B = 6$ some of the valid sequences with the sum N are:

- $[4, 4, 4, 4]$
- $[4, 6, 6]$
- $[6, 4, 6]$
- $[6, 6, 4]$
- $[5, 5, 6]$
- and so on...

For some three integers N, A, and B, he wonders, can N always be expressed as the sum of a valid sequence of integers that fulfilled all the constraints above? If there are multiple sequences, he wants the shortest sequence. If there are multiple sequences with the shortest length, he wants the lexicographically smallest sequence.

Note: A Sequence $X_{1..n}$, of the same length as a sequence $Y_{1..n}$, is considered lexicographically smaller than $Y_{1..n}$ if, and only if, there exists an index j for which X_i = Y_i for $1 \leq i < j$, and $X_j < Y_j$.

So for:

- $N = 4, A = 2, B = 3$, it is possible to find a sequence: $4 = 2 + 2$;
- $N = 10, A = 6, B = 9$, it is impossible find a sequence;
- $N = 251, A = 40, B = 51$, there are multiple sequences, but from all the shortest ones the lexicographically smallest one is $251 = 47 + 51 + 51 + 51 + 51$

Standard input

There are three integers N, A, and B on a single line, separated by single spaces.

Standard output

If it is impossible, output a single line `NO`. Otherwise, output `YES` on the first line. Then output the sequence of integers on the second line, separated by single spaces.

Constraints and notes

- $1 \leq N \leq 10^{15}$
- $1 \leq A \leq B \leq 10^{15}$
- It is guaranteed that if a sequence exists, it does not contain more than 10^5 elements.

Input	Output	Explanation

`4 2 3`	`YES`
	`2 2`
		There is only one solution: `2 2`

`59 8 10`	`YES`
	`9 10 10 10 10 10`
		There are several possible sequences:
		• `10 10 10 10 10 9`
		• `9 10 10 10 10 10`
		• `8 8 8 9 9 9 8`
		• `8 8 8 8 9 9 9`
		• ...
		
		The first two sequences have the shortest length with 6 elements each. The second sequence is lexicographically smaller than the first sequence.

`10 6 9`	`NO`	No sequence exists for this input.

Welcome to IEEEXtreme 13.0

Time limit: *1280 ms*
Memory limit: *264 MB*

On behalf of the executive committee, WELCOME to the 13th IEEEXtreme Programming contest! Our slogan for IEEEXtreme 13.0 is "You're Not Alone" and with over 10,000 participants and volunteers, this has never been truer.

During the next 24 hours, challenges will be released at regular intervals. Keep an eye on the announcements page for updates. It is important that you enjoy this event as much as you can! Hundreds of volunteers have been working for months to prepare this programming party for you! Be sure to get plenty of rest, stay hydrated, and don't forget to eat. Above all, HAVE AS MUCH FUN AS POSSIBLE!

The technical team will be online at all times. We will do our best to answer questions but please note that your code cannot be reviewed under any circumstances. Our first goal is to keep the integrity and fairness of the competition. We encourage you to try to solve every problem, even if your solution is not optimal or even complete. You will be able to obtain a partial score by solving the simple test cases on any given challenge.

All challenges have endured a very rigorous development cycle. We do our best to guarantee their correctness but, in the event you find a mistake, note that the challenges are the same for all contestants. If you have discovered an error, you might want to consider becoming part of our Quality Assurance team for future competitions!

A special thank you to all the volunteers who have made this Xtreme possible. We would like to express our sincerest appreciation to our sponsors and partners for believing in our ideas and supporting this great event.

Your first task is fairly simple, write us a note. Write a program that outputs some text with your feedback and submit it to the system. This task will not be scored and it is not mandatory, however, we look forward to hearing from you!

Good luck and have fun!

Standard input

Nothing on standard input.

Standard output

Output a text with your message to us.

Constraints and notes

- Words ≤ 200