Hello 2019

A. Gennady and a Card Game

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Gennady owns a small hotel in the countryside where he lives a peaceful life. He loves to take long walks, watch sunsets and play cards with tourists staying in his hotel. His favorite game is called "Mau-Mau".

To play Mau-Mau, you need a pack of 52 cards. Each card has a suit (Diamonds — D, Clubs — C, Spades — S, or Hearts — H), and a rank (2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, or A).

At the start of the game, there is one card on the table and you have five cards in your hand. You can play a card from your hand if and only if it has the same rank or the same suit as the card on the table.

In order to check if you'd be a good playing partner, Gennady has prepared a task for you. Given the card on the table and five cards in your hand, check if you can play at least one card.

Input
The first line of the input contains one string which describes the card on the table. The second line contains five strings which describe the cards in your hand.

Each string is two characters long. The first character denotes the rank and belongs to the set $\{2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A\}$. The second character denotes the suit and belongs to the set $\{D, C, S, H\}$.

All the cards in the input are different.

Output
If it is possible to play a card from your hand, print one word "YES". Otherwise, print "NO".

You can print each letter in any case (upper or lower).

Examples

input
AS 2H 4C TH JH AD
output
YES

input
2H 3D 4C AC KD AS
output
NO

input
4D AS AC AD AH 5H
output
YES

Note
In the first example, there is an Ace of Spades (AS) on the table. You can play an Ace of Diamonds (AD) because both of them are Aces.

In the second example, you cannot play any card.

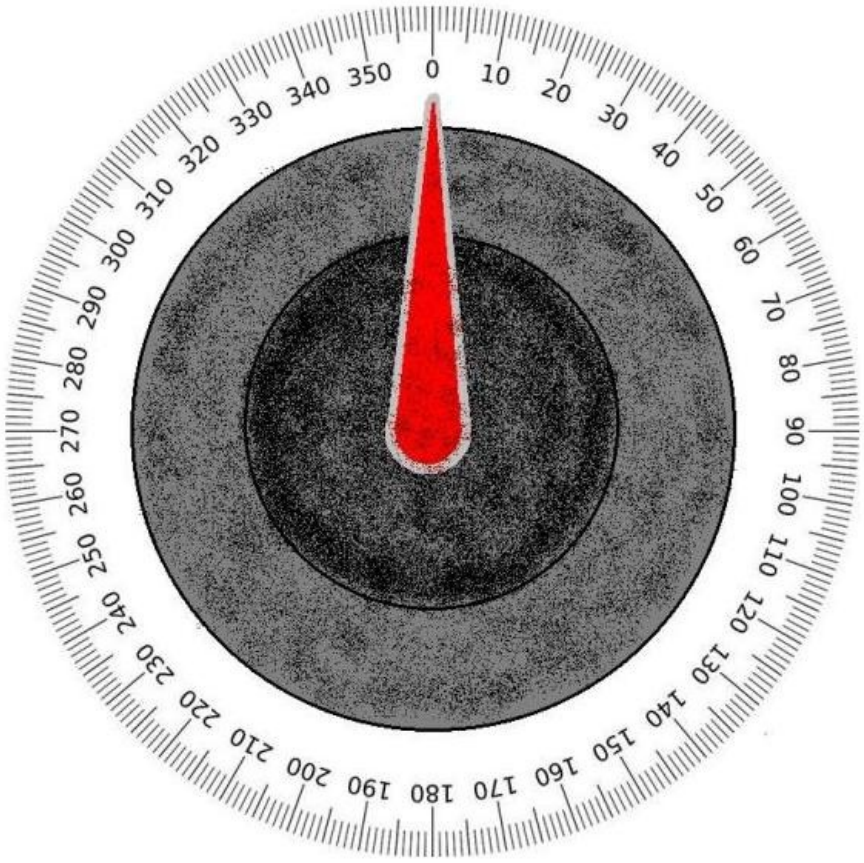
In the third example, you can play an Ace of Diamonds (AD) because it has the same suit as a Four of Diamonds (4D), which lies on the table.

B. Petr and a Combination Lock

time limit per test: 1 second
memory limit per test: 256 megabytes

input: standard input
output: standard output

Petr has just bought a new car. He's just arrived at the most known Petersburg's petrol station to refuel it when he suddenly discovered that the petrol tank is secured with a combination lock! The lock has a scale of 360 degrees and a pointer which initially points at zero:



Petr called his car dealer, who instructed him to rotate the lock's wheel exactly n times. The i-th rotation should be a_i degrees, either clockwise or counterclockwise, and after all n rotations the pointer should again point at zero.

This confused Petr a little bit as he isn't sure which rotations should be done clockwise and which should be done counterclockwise. As there are many possible ways of rotating the lock, help him and find out whether there exists at least one, such that after all n rotations the pointer will point at zero again.

Input

The first line contains one integer n ($1 \le n \le 15$) — the number of rotations.

Each of the following n lines contains one integer a_i ($1 \le a_i \le 180$) — the angle of the i-th rotation in degrees.

Output

If it is possible to do all the rotations so that the pointer will point at zero after all of them are performed, print a single word "YES". Otherwise, print "NO". Petr will probably buy a new car in this case.

You can print each letter in any case (upper or lower).

Examples

input
3 10 20 30
output
YES

input
3 10 10 10
output
NO

input
3 120

120 120

output
YES

Note
In the first example, we can achieve our goal by applying the first and the second rotation clockwise, and performing the third rotation counterclockwise.

In the second example, it's impossible to perform the rotations in order to make the pointer point at zero in the end.

In the third example, Petr can do all three rotations clockwise. In this case, the whole wheel will be rotated by 360 degrees clockwise and the pointer will point at zero again.

C. Yuhao and a Parenthesis

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One day, Yuhao came across a problem about checking if some bracket sequences are correct bracket sequences.

A bracket sequence is any non-empty sequence of opening and closing parentheses. A bracket sequence is called a *correct bracket sequence* if it's possible to obtain a correct arithmetic expression by inserting characters "+" and "1" into this sequence. For example, the sequences "(( )) ( )", "( )" and "(( ) (( )))" are correct, while the bracket sequences ") (", "(( )" and "(( ))) (" are not correct.

Yuhao found this problem too simple for him so he decided to make the problem harder. You are given many (not necessarily correct) bracket sequences. The task is to connect some of them into ordered pairs so that each bracket sequence occurs in at most one pair and the concatenation of the bracket sequences in each pair is a correct bracket sequence. The goal is to create as many pairs as possible.

This problem unfortunately turned out to be too difficult for Yuhao. Can you help him and solve it?

Input
The first line contains one integer n ($1 \leq n \leq 10^5$) — the number of bracket sequences.

Each of the following n lines contains one bracket sequence — a non-empty string which consists only of characters "(" and ")".

The sum of lengths of all bracket sequences in the input is at most $5 \cdot 10^5$.

Note that a bracket sequence may appear in the input multiple times. In this case, you can use each copy of the sequence separately. Also note that the order in which strings appear in the input doesn't matter.

Output
Print a single integer — the maximum number of pairs which can be made, adhering to the conditions in the statement.

Examples

input
7)())) ((((( ))

output
2

input
4 (((((((())

output
0

input
2 (()) ()

output

1

Note
In the first example, it's optimal to construct two pairs: "(( ) ( ) ( ))" and "(( ))".

D. Makoto and a Blackboard

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Makoto has a big blackboard with a positive integer n written on it. He will perform the following action exactly k times:

Suppose the number currently written on the blackboard is v. He will randomly pick one of the divisors of v (possibly 1 and v) and replace v with this divisor. As Makoto uses his famous random number generator (RNG) and as he always uses 58 as his generator seed, each divisor is guaranteed to be chosen with equal probability.

He now wonders what is the expected value of the number written on the blackboard after k steps.

It can be shown that this value can be represented as $\frac{P}{Q}$ where P and Q are coprime integers and $Q \not\equiv 0 \pmod{10^9+7}$. Print the value of $P \cdot Q^{-1}$ modulo 10^9+7.

Input
The only line of the input contains two integers n and k ($1 \le n \le 10^{15}$, $1 \le k \le 10^4$).

Output
Print a single integer — the expected value of the number on the blackboard after k steps as $P \cdot Q^{-1} \pmod{10^9+7}$ for P, Q defined above.

Examples

input
6 1
output
3

input
6 2
output
875000008

input
60 5
output
237178099

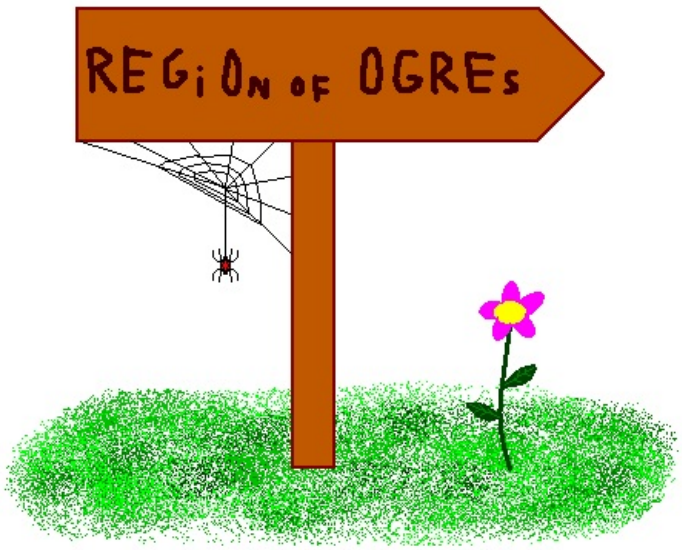
Note
In the first example, after one step, the number written on the blackboard is 1, 2, 3 or 6 — each occurring with equal probability. Hence, the answer is $\frac{1+2+3+6}{4} = 3$.

In the second example, the answer is equal to $1 \cdot \frac{9}{16} + 2 \cdot \frac{3}{16} + 3 \cdot \frac{3}{16} + 6 \cdot \frac{1}{16} = \frac{15}{8}$.

E. Egor and an RPG game

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One Saturday afternoon Egor was playing his favorite RPG game. While discovering new lands and territories, he came across the following sign:



Egor is a passionate player, but he is an algorithmician as well. That's why he instantly spotted four common letters in two words on the sign above — if we *permute* the letters "R", "E", "G", "O" from the first word, we can obtain the letters "O", "G", "R", "E". Egor got inspired by the sign and right away he came up with a problem about permutations.

You are given a permutation of length n. You have to split it into some non-empty subsequences so that each element of the permutation belongs to exactly one subsequence. Each subsequence must be *monotonic* — that is, either increasing or decreasing.

Sequence is called to be a *subsequence* if it can be derived from permutation by deleting some (possibly none) elements without changing the order of the remaining elements.

The number of subsequences should be small enough — let $f(n)$ be the minimum integer k such that every permutation of length n can be partitioned into at most k monotonic subsequences.

You need to split the permutation into at most $f(n)$ monotonic subsequences.

Input

The first line contains one integer t ($1 \le t \le 10^5$) — the number of test cases.

You can only use $t = 1$ in hacks.

Next, descriptions of t test cases come, each of them in the following format.

The first line of a single test case contains one integer n ($1 \le n \le 10^5$) — the length of the permutation. The second line contains n distinct integers a_i ($1 \le a_i \le n$) — the permutation itself.

The sum of the values of n over all test cases doesn't exceed 10^5.

Output

For each test case print the answer in the following format:

In the first line print k ($1 \le k \le f(n)$) — the number of the subsequences in the partition. In the next k lines, print the descriptions of found subsequences. Each description should start with a number l_i ($1 \le l_i \le n$) — the length of the corresponding subsequence, followed by l_i integers — the values of this subsequence in the order in which they occur in the permutation.

Each subsequence you output must be either increasing or decreasing.

In case there are multiple possible answers, print any of them.

Example

input
3 4 4 3 1 2 6 4 5 6 1 3 2 10 1 2 3 4 5 6 7 8 9 10
output
2 3 4 3 1 1 2 3 2 4 1 2 5 6 2 3 2 1 10 1 2 3 4 5 6 7 8 9 10

Note

In the example, we can split:

- $[4, 3, 1, 2]$ into $[4, 3, 1]$, $[2]$
- $[4, 5, 6, 1, 3, 2]$ into $[4, 1]$, $[5, 6]$ and $[3, 2]$
- $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ into $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

Surely, there are many more answers possible.

F. Alex and a TV Show

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alex decided to try his luck in TV shows. He once went to the quiz named "*What's That Word?!*". After perfectly answering the questions "*How is a pseudonym commonly referred to in the Internet?*" ("Um... a *nick*?"), *"After which famous inventor we name the unit of the magnetic field strength?"* ("Um... Nikola Tesla?") and "*Which rock band performs "How You Remind Me"?*" ("Um... Nickelback?"), he decided to apply to a little bit more difficult TV show: "What's in This Multiset?!".

The rules of this TV show are as follows: there are n multisets numbered from 1 to n. Each of them is initially empty. Then, q events happen; each of them is in one of the four possible types:

- 1 x v — set the x-th multiset to a singleton $\{v\}$.
- 2 x y z — set the x-th multiset to a union of the y-th and the z-th multiset. For example: $\{1, 3\} \cup \{1, 4, 4\} = \{1, 1, 3, 4, 4\}$.
- 3 x y z — set the x-th multiset to a product of the y-th and the z-th multiset. The product $A \times B$ of two multisets A, B is defined as $\{\gcd(a, b) \mid a \in A,\ b \in B\}$, where $\gcd(p, q)$ is the greatest common divisor of p and q. For example: $\{2, 2, 3\} \times \{1, 4, 6\} = \{1, 2, 2, 1, 2, 2, 1, 1, 3\}$.
- 4 x v — the participant is asked how many times number v occurs in the x-th multiset. As the quiz turned out to be too hard in the past, participants should now give the answers **modulo 2 only**.

Note, that x, y and z described above are not necessarily different. In events of types 2 and 3, the sum or the product is computed first, and then the assignment is performed.

Alex is confused by the complicated rules of the show. Can you help him answer the requests of the 4-th type?

Input

The first line contains two integers n and q ($1 \le n \le 10^5$, $1 \le q \le 10^6$) — the number of multisets and the number of events.

Each of the following q lines describes next event in the format given in statement. It's guaranteed that $1 \le x, y, z \le n$ and $1 \le v \le 7000$ always holds.

It's guaranteed that there will be at least one event of the 4-th type.

Output

Print a string which consists of digits 0 and 1 only, and has length equal to the number of events of the 4-th type. The i-th digit of the string should be equal to the answer for the i-th query of the 4-th type.

Example

input
4 13 1 1 1 1 2 4 1 3 6 4 4 4 1 4 4 2 2 1 2 2 3 3 4 4 4 4 3 2 2 3 4 2 1 4 2 2 4 2 3 4 2 4
output
010101

Note

Here is how the multisets look in the example test after each of the events; i is the number of queries processed so far:

i	first	second	third	fourth
0	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
1	$\{1\}$	$\varnothing$	$\varnothing$	$\varnothing$
2	$\{1\}$	$\{4\}$	$\varnothing$	$\varnothing$
3	$\{1\}$	$\{4\}$	$\{6\}$	$\varnothing$
4	$\{1\}$	$\{4\}$	$\{6\}$	$\varnothing$
5	$\{1\}$	$\{4\}$	$\{6\}$	$\{4\}$
6	$\{1\}$	$\{1,4\}$	$\{6\}$	$\{4\}$
7	$\{1\}$	$\{1,4\}$	$\{4,6\}$	$\{4\}$
8	$\{1\}$	$\{1,4\}$	$\{4,6\}$	$\{4\}$
9	$\{1\}$	$\{1,1,2,4\}$	$\{4,6\}$	$\{4\}$
10	$\{1\}$	$\{1,1,2,4\}$	$\{4,6\}$	$\{4\}$
11	$\{1\}$	$\{1,1,2,4\}$	$\{4,6\}$	$\{4\}$
12	$\{1\}$	$\{1,1,2,4\}$	$\{4,6\}$	$\{4\}$
13	$\{1\}$	$\{1,1,2,4\}$	$\{4,6\}$	$\{4\}$

G. Vladislav and a Great Legend

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A great legend used to be here, but some troll hacked Codeforces and erased it. Too bad for us, but in the troll society he earned a title of an ultimate-greatest-over troll. At least for them, it's something good. And maybe a formal statement will be even better for us?

You are given a tree T with n vertices numbered from 1 to n. For every **non-empty** subset X of vertices of T, let $f(X)$ be the minimum number of edges in the smallest connected subtree of T which contains every vertex from X.

You're also given an integer k. You need to compute the sum of $(f(X))^k$ among all non-empty subsets of vertices, that is:

$$\sum_{X \subseteq \{1,2,\ldots,n\},\, X \neq \varnothing} (f(X))^k.$$

As the result might be very large, output it modulo $10^9 + 7$.

Input

The first line contains two integers n and k ($2 \le n \le 10^5$, $1 \le k \le 200$) — the size of the tree and the exponent in the sum above.

Each of the following $n - 1$ lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$) — the indices of the vertices connected by the corresponding edge.

It is guaranteed, that the edges form a tree.

Output

Print a single integer — the requested sum modulo $10^9 + 7$.

Examples

input
4 1 1 2 2 3 2 4
output
21

input
4 2 1 2 2 3 2 4
output
45

input

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

output

```
780
```

Note

In the first two examples, the values of f are as follows:

$f(\{1\}) = 0$

$f(\{2\}) = 0$

$f(\{1, 2\}) = 1$

$f(\{3\}) = 0$

$f(\{1, 3\}) = 2$

$f(\{2, 3\}) = 1$

$f(\{1, 2, 3\}) = 2$

$f(\{4\}) = 0$

$f(\{1, 4\}) = 2$

$f(\{2, 4\}) = 1$

$f(\{1, 2, 4\}) = 2$

$f(\{3, 4\}) = 2$

$f(\{1, 3, 4\}) = 3$

$f(\{2, 3, 4\}) = 2$

$f(\{1, 2, 3, 4\}) = 3$

H. Mateusz and an Infinite Sequence

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

A *Thue-Morse-Radecki-Mateusz sequence* (Thorse-Radewoosh sequence in short) is an infinite sequence constructed from a finite sequence gen of length d and an integer m, obtained in the following sequence of steps:

- In the beginning, we define the one-element sequence $M_0 = (0)$.
- In the k-th step, $k \geq 1$, we define the sequence M_k to be the concatenation of the d copies of M_{k-1}. However, each of them is altered slightly — in the i-th of them ($1 \leq i \leq d$), each element x is changed to $(x + \mathrm{gen}_i) \pmod m$.

For instance, if we pick $\mathrm{gen} = (0, 1, 2)$ and $m = 4$:

- $M_0 = (0)$,
- $M_1 = (0, 1, 2)$,
- $M_2 = (0, 1, 2, 1, 2, 3, 2, 3, 0)$,
- $M_3 = (0, 1, 2, 1, 2, 3, 2, 3, 0, 1, 2, 3, 2, 3, 0, 3, 0, 1, 2, 3, 0, 3, 0, 1, 0, 1, 2)$, and so on.

As you can see, as long as the first element of gen is 0, each consecutive step produces a sequence whose prefix is the sequence generated in the previous step. Therefore, we can define the infinite Thorse-Radewoosh sequence M_∞ as the sequence obtained by applying the step above indefinitely. For the parameters above, $M_\infty = (0, 1, 2, 1, 2, 3, 2, 3, 0, 1, 2, 3, 2, 3, 0, 3, 0, 1, \ldots)$.

Mateusz picked a sequence gen and an integer m, and used them to obtain a Thorse-Radewoosh sequence M_∞. He then picked two integers l, r, and wrote down a subsequence of this sequence $A := ((M_\infty)_l, (M_\infty)_{l+1}, \ldots, (M_\infty)_r)$.

Note that we use the 1-based indexing both for M_∞ and A.

Mateusz has his favorite sequence B with length n, and would like to see how large it is compared to A. Let's say that B majorizes sequence X of length n (let's denote it as $B \geq X$) if and only if for all $i \in \{1, 2, \ldots, n\}$, we have $B_i \geq X_i$.

He now asks himself how many integers x in the range $[1, |A| - n + 1]$ there are such that $B \geq (A_x, A_{x+1}, A_{x+2}, \ldots, A_{x+n-1})$. As both sequences were huge, answering the question using only his pen and paper turned out to be too time-consuming. Can you help him automate his research?

Input

The first line contains two integers d and m ($2 \le d \le 20$, $2 \le m \le 60$) — the length of the sequence gen and an integer used to perform the modular operations. The second line contains d integers gen_i ($0 \le \mathrm{gen}_i < m$). It's guaranteed that the first element of the sequence gen is equal to zero.

The third line contains one integer n ($1 \le n \le 30000$) — the length of the sequence B. The fourth line contains n integers B_i ($0 \le B_i < m$). The fifth line contains two integers l and r ($1 \le l \le r \le 10^{18}$, $r - l + 1 \ge n$).

Output

Print a single integer — the answer to the problem.

Examples

input
2 2 0 1 4 0 1 1 0 2 21
output
6

input
3 4 0 1 2 2 0 2 6 11
output
1

Note

Thorse-Radewoosh sequence in the first example is the standard Thue-Morse sequence, so the sequence A is as follows: 11010011001011010010. Here are the places where the sequence B majorizes A: