A. Bovine Dilemma

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

Argus was charged with guarding Io, which is not an ordinary cow. Io is quite an explorer, and she wanders off rather frequently, making Argus' life stressful. So the cowherd decided to construct an enclosed pasture for Io.

There are n trees growing along the river, where Argus tends Io. For this problem, the river can be viewed as the OX axis of the Cartesian coordinate system, and the n trees as points with the y-coordinate equal 0. There is also another tree growing in the point $(0, 1)$.

Argus will tie a rope around three of the trees, creating a triangular pasture. Its exact shape doesn't matter to Io, but its area is crucial to her. There may be many ways for Argus to arrange the fence, but only the ones which result in different areas of the pasture are interesting for Io. Calculate the number of **different** areas that her pasture may have. Note that the pasture must have **nonzero** area.

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow, each one is described in two lines.

In the first line of each test case there is a single integer n ($1 \le n \le 50$) denoting the number of trees growing along the river. Next line contains n distinct integers $x_1 < x_2 < \ldots < x_{n-1} < x_n$ ($1 \le x_i \le 50$), the x-coordinates of trees growing along the river.

Output

In a single line output an integer, the number of **different nonzero** areas that triangles with trees as vertices may have.

Example

input
8 4 1 2 4 5 3 1 3 5 3 2 6 8 2 1 2 1 50 5 3 4 5 6 8 3 1 25 26 6 1 2 4 8 16 32

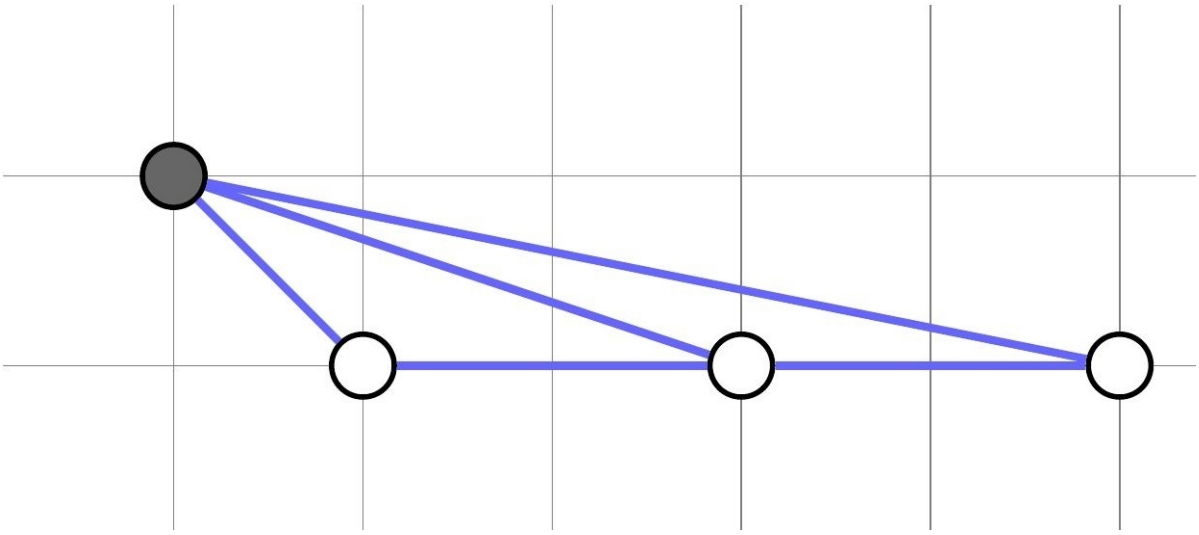
output
4 2 3 1 0 5 3 15

Note

In the first test case, we have 6 non-degenerate triangles with the following areas: 0.5, 0.5, 1, 1.5, 1.5 and 2. The pasture can have 4 different areas, and thus 4 is the answer.

In the second test case, we have 3 non-degenerate triangles with the following areas: 1, 1 and 2. The pasture can have 2 different areas, so 2 is the answer.

The following two drawings present the situation in the second test case. The blue triangles in the first drawing have area 1. The red triangle in the second drawing has area 2.



B. Last minute enhancements

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

Athenaeus has just finished creating his latest musical composition and will present it tomorrow to the people of Athens. Unfortunately, the melody is rather dull and highly likely won't be met with a warm reception.

His song consists of n notes, which we will treat as **positive integers**. The **diversity** of a song is the number of **different** notes it contains. As a patron of music, Euterpe watches over composers and guides them throughout the process of creating new melodies. She decided to help Athenaeus by changing his song to make it more diverse.

Being a minor goddess, she cannot arbitrarily change the song. Instead, for each of the n notes in the song, she can either leave it as it is or **increase** it by 1.

Given the song as a sequence of integers describing the notes, find out the maximal, achievable diversity.

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 10\,000$) — the number of test cases. Then t test cases follow, each one is described in two lines.

In the first line of each test case there is a single integer n ($1 \le n \le 10^5$) denoting the length of the song. The next line contains a sequence of n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_1 \le x_2 \le \ldots \le x_n \le 2 \cdot n$), describing the song.

The sum of n over all test cases does not exceed 10^5.

Output
For each test case, you should output a single line containing precisely one integer, the maximal diversity of the song, i.e. the maximal possible number of different elements in the final sequence.

Example

input
5 6 1 2 2 2 5 6 2 4 4 6 1 1 3 4 4 5 1 1 6 1 1 1 2 2 2

output
5 2 6 1 3

Note
In the first test case, Euterpe can increase the second, fifth and sixth element to obtain the sequence $1, \underline{3}, 2, 2, \underline{6}, \underline{7}$, which has 5 different elements (increased elements are underlined).

In the second test case, Euterpe can increase the first element to obtain the sequence $\underline{5}, 4$, which has 2 different elements.

In the third test case, Euterpe can increase the second, fifth and sixth element to obtain the sequence $1, \underline{2}, 3, 4, \underline{5}, \underline{6}$, which has 6 different elements.

C. Canine poetry

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

After his wife's tragic death, Eurydice, Orpheus descended to the realm of death to see her. Reaching its gates was uneasy, but passing through them proved to be even more challenging. Mostly because of Cerberus, the three-headed hound of Hades.

Orpheus, a famous poet, and musician plans to calm Cerberus with his poetry and safely walk past him. He created a very peculiar poem for Cerberus. It consists only of lowercase English letters.

We call a poem's substring a palindrome if and only if it reads the same backwards and forwards. A string a is a substring of a string b if a can be obtained from b by deleting several (possibly zero or all) characters from the beginning and several (possibly zero or all) characters from the end.

Unfortunately, Cerberus dislikes palindromes of length greater than 1. For example in the poem abaa the hound of Hades wouldn't like substrings aba and aa.

Orpheus can only calm Cerberus if the hound likes his poetry. That's why he wants to change his poem so that it does not contain any palindrome substrings of length **greater than** 1.

Orpheus can modify the poem by replacing a letter at any position with any lowercase English letter. He can use this operation arbitrarily many times (possibly zero). Since there can be many palindromes in his poem, he may have to make some corrections. But how many, exactly? Given the poem, determine the minimal number of letters that have to be changed so that the poem does not contain any palindromes of length greater than 1.

Input
The first line of the input contains a single integer t ($1 \le t \le 10^5$) denoting the number of test cases, then t test cases follow.

The first and only line of each test case contains a non-empty string of lowercase English letters, Orpheus' poem.

The sum of the length of Orpheus' poems in all test cases will not exceed 10^5.

Output
You should output t lines, i-th line should contain a single integer, answer to the i-th test case.

Example

input
7 babba abaac codeforces zeroorez abcdcba bbbbbbb a

output
1 1 0 1 1 4 0

Note
In the first test case, we can replace the third character with c and obtain a palindrome-less poem bacba.

In the second test case, we can replace the third character with d and obtain a palindrome-less poem abdac.

In the third test case, the initial poem already doesn't contain any palindromes, so Orpheus doesn't need to change anything there.

D. 13th Labour of Heracles

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You've probably heard about the twelve labors of Heracles, but do you have any idea about the thirteenth? It is commonly assumed it took him a dozen years to complete the twelve feats, so on average, a year to accomplish every one of them. As time flows faster these days, you have minutes rather than months to solve this task. But will you manage?

In this problem, you are given a tree with n weighted vertices. A tree is a connected graph with $n - 1$ edges.

Let us define its k-coloring as an assignment of k colors to the edges so that each edge has exactly one color assigned to it. Note that you don't have to use all k colors.

A subgraph of color x consists of these edges from the original tree, which are assigned color x, and only those vertices that are adjacent to at least one such edge. So there are no vertices of degree 0 in such a subgraph.

The value of a connected component is the **sum** of weights of its vertices. Let us define the value of a subgraph as a **maximum** of values of its connected components. We will assume that the value of an empty subgraph equals 0.

There is also a value of a k-coloring, which equals the **sum** of values of subgraphs of all k colors. Given a tree, for each k from 1 to $n-1$ calculate the maximal value of a k-coloring.

Input

In the first line of input, there is a single integer t ($1 \le t \le 10^5$) denoting the number of test cases. Then t test cases follow.

First line of each test case contains a single integer n ($2 \le n \le 10^5$). The second line consists of n integers $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^9$). w_i equals the weight of i-th vertex. In each of the following $n-1$ lines, there are two integers u, v ($1 \le u, v \le n$) describing an edge between vertices u and v. It is guaranteed that these edges form a tree.

The sum of n in all test cases will not exceed $2 \cdot 10^5$.

Output

For every test case, your program should print one line containing $n-1$ integers separated with a single space. The i-th number in a line should be the maximal value of a i-coloring of the tree.

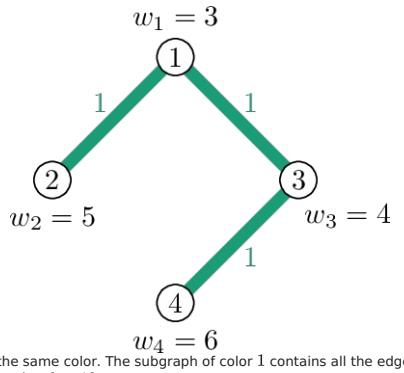
Example

input
4 4 3 5 4 6 2 1 3 1 4 3 2 21 32 2 1 6 20 13 17 13 13 11 2 1 3 1 4 1 5 1 6 1 4 10 6 6 6 1 2 2 3 4 1

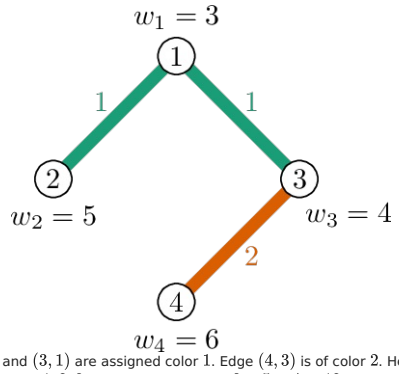
output
18 22 25 53 87 107 127 147 167 28 38 44

Note

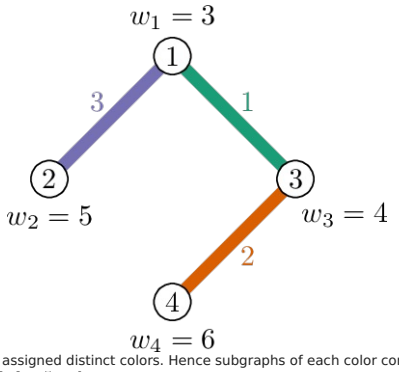
The optimal k-colorings from the first test case are the following:



In the 1-coloring all edges are given the same color. The subgraph of color 1 contains all the edges and vertices from the original graph. Hence, its value equals $3 + 5 + 4 + 6 = 18$.



In an optimal 2-coloring edges $(2, 1)$ and $(3, 1)$ are assigned color 1. Edge $(4, 3)$ is of color 2. Hence the subgraph of color 1 consists of a single connected component (vertices $1, 2, 3$) and its value equals $3 + 5 + 4 = 12$. The subgraph of color 2 contains two vertices and one edge. Its value equals $4 + 6 = 10$.



In an optimal 3-coloring all edges are assigned distinct colors. Hence subgraphs of each color consist of a single edge. They values are as follows: $3 + 4 = 7$, $4 + 6 = 10$, $3 + 5 = 8$.

E. Apollo versus Pan

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

Only a few know that Pan and Apollo weren't only battling for the title of the GOAT musician. A few millenniums later, they also challenged each other in math (or rather in fast calculations). The task they got to solve is the following:

Let $x_1, x_2, \ldots, x_n$ be the sequence of n non-negative integers. Find this value:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(x_i \,\&\, x_j) \cdot (x_j \mid x_k)$$

Here $\&$ denotes the bitwise and, and $\mid$ denotes the bitwise or.

Pan and Apollo could solve this in a few seconds. Can you do it too? For convenience, find the answer modulo $10^9 + 7$.

Input
The first line of the input contains a single integer t ($1 \le t \le 1\,000$) denoting the number of test cases, then t test cases follow.

The first line of each test case consists of a single integer n ($1 \le n \le 5 \cdot 10^5$), the length of the sequence. The second one contains n non-negative integers $x_1, x_2, \ldots, x_n$ ($0 \le x_i < 2^{60}$), elements of the sequence.

The sum of n over all test cases will not exceed $5 \cdot 10^5$.

Output
Print t lines. The i-th line should contain the answer to the i-th text case.

Example

input
8 2 1 7 3 1 2 4 4 5 5 5 5 5 6 2 2 1 0 1 0 1 1 6 1 12 123 1234 12345 123456 5 536870912 536870911 1152921504606846975 1152921504606846974 1152921504606846973

output
128 91 1600 505 0 1 502811676 264880351

F. Euclid's nightmare

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

You may know that Euclid was a mathematician. Well, as it turns out, Morpheus knew it too. So when he wanted to play a mean trick on Euclid, he sent him an appropriate nightmare.

In his bad dream Euclid has a set S of n m-dimensional vectors over the $\mathbb{Z}_2$ field and can perform vector addition on them. In other words he has vectors with m coordinates, each one equal either 0 or 1. Vector addition is defined as follows: let $u + v = w$, then $w_i = (u_i + v_i) \bmod 2$.

Euclid can sum any subset of S and archive another m-dimensional vector over $\mathbb{Z}_2$. In particular, he can sum together an empty subset; in such a case, the resulting vector has all coordinates equal 0.

Let T be the set of all the vectors that can be written as a sum of some vectors from S. Now Euclid wonders the size of T and whether he can use only a subset S' of S to obtain all the vectors from T. As it is usually the case in such scenarios, he will not wake up until he figures this out. So far, things are looking rather grim for the philosopher. But there is hope, as he noticed that all vectors in S have **at most** 2 coordinates equal 1.

Help Euclid and calculate $|T|$, the number of m-dimensional vectors over $\mathbb{Z}_2$ that can be written as a sum of some vectors from S. As it can be quite large, calculate it modulo $10^9 + 7$. You should also find S', the **smallest** such subset of S, that all vectors in T can be written as a sum of vectors from S'. In case there are multiple such sets with a minimal number of elements, output the lexicographically smallest one with respect to the order in which their elements are given in the input.

Consider sets A and B such that $|A| = |B|$. Let $a_1, a_2, \ldots a_{|A|}$ and $b_1, b_2, \ldots b_{|B|}$ be increasing arrays of indices elements of A and B correspondingly. A is lexicographically smaller than B iff there exists such i that $a_j = b_j$ for all $j < i$ and $a_i < b_i$.

Input
In the first line of input, there are two integers n, m ($1 \le n, m \le 5 \cdot 10^5$) denoting the number of vectors in S and the number of dimensions.

Next n lines contain the description of the vectors in S. In each of them there is an integer k ($1 \le k \le 2$) and then follow k distinct integers $x_1, \ldots x_k$ ($1 \le x_i \le m$). This encodes an m-dimensional vector having 1s on coordinates $x_1, \ldots x_k$ and 0s on the rest of them.

Among the n vectors, no two are the same.

Output
In the first line, output two integers: remainder modulo $10^9 + 7$ of $|T|$ and $|S'|$. In the second line, output $|S'|$ numbers, indices of the elements of S' in ascending order. The elements of S are numbered from 1 in the order they are given in the input.

Examples

input
3 2 1 1 1 2 2 2 1

output
4 2 1 2

input
2 3 2 1 3 2 1 2

output
4 2 1 2

input
3 5 2 1 2 1 3 1 4

output
8 3 1 2 3

Note
In the first example we are given three vectors:

- 10
- 01
- 11

It turns out that we can represent all vectors from our 2-dimensional space using these vectors:

- 00 is a sum of the empty subset of above vectors;
- $01 = 11 + 10$, is a sum of the first and third vector;
- $10 = 10$, is just the first vector;
- $11 = 10 + 01$, is a sum of the first and the second vector.

Hence, $T = \{00, 01, 10, 11\}$. We can choose any two of the three vectors from S and still be able to obtain all the vectors in T. In such a case, we choose the two vectors which appear first in the input. Since we cannot obtain all vectors in T using only a single vector from S, $|S'| = 2$ and $S' = \{10, 01\}$ (indices 1 and 2), as set $\{1, 2\}$ is lexicographically the smallest. We can represent all vectors from T, using only vectors from S', as shown below:

- 00 is a sum of the empty subset;
- $01 = 01$ is just the second vector;
- $10 = 10$ is just the first vector;
- $11 = 10 + 01$ is a sum of the first and the second vector.

G. Song of the Sirens

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

Whoso in ignorance draws near to them and hears the Sirens' voice, he nevermore returns.
Homer, *Odyssey*

In the times of Jason and the Argonauts, it was well known that sirens use the sound of their songs to lure sailors into their demise. Yet only a few knew that every time sirens call a sailor by his name, his will weakens, making him more vulnerable.

For the purpose of this problem, both siren songs and names of the sailors will be represented as strings of lowercase English letters. The more times the sailor's name occurs as a contiguous substring of the song, the greater danger he is in.

Jason found out that sirens can sing one of the $n + 1$ songs, which have the following structure: let s_i ($0 \le i \le n$) be the i-th song and t be a string of length n, then for every $i < n$: $s_{i+1} = s_i t_i s_i$. In other words $i + 1$-st song is the concatenation of i-th song, i-th letter (0-indexed) of t and the i-th song.

Fortunately, he also knows s_0 and t. Jason wonders how many times a sailor's name is mentioned in a particular song. Answer q queries: given the sailor's name (w) and the index of a song (i) output the number of occurrences of w in s_i as a substring. As this number can be quite large, output its remainder modulo $10^9 + 7$.

Input

In the first line of input there are two integers n, q ($1 \le n, q \le 10^5$) meaning that there are $n + 1$ songs and q queries. In the next two lines strings s_0 and t follow ($1 \le |s_0| \le 100, |t| = n$).

Next q lines describe the queries; each one contains an integer k ($0 \le k \le n$), the index of the song sung by the sirens, and a non-empty string w, which is the name of a sailor. All strings in this problem consist only of lowercase English letters, and the sum of lengths of sailors' names does not exceed 10^6.

Output

Output q lines, i-th of them should contain the remainder modulo $10^9 + 7$ of the number of occurrences of w in s_k.

Examples

input
3 3 aa bcd 2 aba 3 ca 3 aa
output
2 2 8

input
4 5 aba bbac 1 a 3 baca 3 ab 2 bab 4 aba
output
4 0 14 6 28

Note

In the first example songs of the sirens are as follows:

- Song 0: aa
- Song 1: aabaa
- Song 2: aabaacaabaa
- Song 3: aabaacaabaadaabaacaabaa

H. Finding satisfactory solutions

time limit per test: 5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Getting so far in this contest is not an easy feat. By solving all the previous problems, you have impressed the gods greatly. Thus, they decided to spare you the story for this problem and grant a formal statement instead.

Consider n agents. Each one of them initially has exactly one item, **i-th agent has the item number** i. We are interested in reassignments of these items among the agents. An assignment is valid iff each item is assigned to exactly one agent, and each agent is assigned exactly one item.

Each agent has a preference over the items, which can be described by a permutation p of items sorted from the most to the least desirable. In other words, the agent prefers item i to item j iff i appears earlier in the permutation p. A **preference profile** is a list of n permutations of length n each, such that i-th permutation describes preferences of the i-th agent.

It is possible that some of the agents are not happy with the assignment of items. A set of dissatisfied agents may choose not to cooperate with other agents. In such a case, they would exchange the items they possess initially (i-th item belongs to i-th agent) only between themselves. Agents from this group don't care about the satisfaction of agents outside of it. However, they need to exchange their items in such a way that will make at least one of them happier, and none of them less happy (in comparison to the given assignment).

Formally, consider a valid assignment of items — A. Let $A(i)$ denote the item assigned to i-th agent. Also, consider a subset of agents. Let S be the set of their indices. We will say this subset of agents is dissatisfied iff there exists a valid assignment $B(i)$ such

that:

- For each $i \in S$, $B(i) \in S$.
- No agent $i \in S$ prefers $A(i)$ to $B(i)$ (no agent from the S is less happy).
- At least one agent $i \in S$ prefers $B(i)$ to $A(i)$ (at least one agent from the S is happier).

An assignment is optimal if no subset of the agents is dissatisfied. Note that the empty subset cannot be dissatisfied. It can be proven that for each preference profile, there is precisely one optimal assignment.

Example: Consider 3 agents with the following preference profile:

1. $[2, 1, 3]$
2. $[1, 2, 3]$
3. $[1, 3, 2]$

And such an assignment:

- First agent gets item 2
- Second agent gets item 3.
- Third agent gets item 1.

See that the set of agents $\{1, 2\}$ is dissatisfied, because they can reassign their (initial) items in the following way:

- First agent gets item 2.
- Second agent gets item 1.
- Third agent gets item 3.

This reassignment will make the second agent happier and make no difference to the first agent. As a result, the third agent got an item that is worse for him, but this does not prevent the set $\{1, 2\}$ from being dissatisfied (he is not in this set).

The following assignment would be optimal:

- First agent gets item 2.
- Second agent gets item 1.
- Third agent gets item 3.

Given an assignment A, calculate the number of distinct preference profiles for which assignment A is optimal. As the answer can be huge, output it modulo $10^9 + 7$.

Two preference profiles are different iff they assign different preference permutations to any agent.

Input
In the first line of input there is an integer n ($1 \leq n \leq 40$). The next line contains n space separated integers, a permutation of numbers from 1 to n. The i-th number denotes the item assigned to agent i in the optimal assignment.

Output
In a single line output one non-negative integer, the number of preference profiles for which the assignment of items given in the input is optimal modulo $10^9 + 7$.

Examples

input
2 2 1
output
1

input
3 1 2 3
output
98

input
4 2 1 3 4
output
27408

Note
Assignment from the first test case is optimal only for the following preference profile:

$2, 1$

$1, 2$

If any agent wants his initial item the most and is given another item, he would form a dissatisfied set. Hence the allocation is not optimal for any other preference profile.

I. The Riddle of the Sphinx

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

What walks on four feet in the morning, two in the afternoon, and three at night?

This is an interactive problem. This problem doesn't support hacks.

Sphinx's duty is to guard the city of Thebes by making sure that no unworthy traveler crosses its gates. Only the ones who answer her riddle timely and correctly (or get an acc for short) are allowed to pass. As of those who fail, no one heard of them ever again...

So you don't have a choice but to solve the riddle. Sphinx has an array $a_1, a_2, \ldots, a_n$ of **nonnegative** integers **strictly smaller** than 2^b and asked you to find the maximum value among its elements. Of course, she will not show you the array, but she will give you n and b. As it is impossible to answer this riddle blindly, you can ask her some questions. For given i, y, she'll answer you whether a_i is **bigger** than y. As sphinxes are not very patient, you can ask at most $3 \cdot (n + b)$ such questions.

Although cunning, sphinxes are honest. Even though the array **can change** between your queries, answers to the previously asked questions will remain valid.

Input
The first line contains two integers n and b ($1 \leq n, b \leq 200$). The remaining parts of the input will be given throughout the interaction process.

Interaction
In each round your program must output a single line with an integer i ($0 \leq i \leq n$) and a binary string of length **exactly** b denoting the **binary representation** of y (most significant bit first).

If $i > 0$, this line encodes the question: *Is a_i bigger than y?*. There should be at most $3 \cdot (n + b)$ such lines; after each of them, the interactor will print `yes` or `no` in a single line.

If $i = 0$, this is the last round of interaction, after which your program should terminate, and y should be the maximal value among the elements of Sphinx's array. Note that this round **does not** count to the query limit.

Note that the interactor **is adaptive**.

After printing a query, do not forget to output the end of the line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;

- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

If your solution does not correctly follow the interaction guideline above, it may receive an arbitrary verdict. Otherwise, your program will receive the `Wrong Answer` judgment if it reports the wrong maximum.

Examples

input
5 3 yes no no no no yes
output
5 101 5 110 4 100 3 101 2 001 1 000 0 110

input
4 3 no no no no
output
1 000 2 000 3 000 4 000 0 000

input
1 1
output
0 0

Note

In all examples, the sequence is **fixed** beforehand.

In the first example, the sequence is $2, 1, 4, 0, 6$.

In the second example, the sequence is $0, 0, 0, 0$.

In the third example, the sequence is 0.

Note that if the interactor was adaptive, then the interaction in the first and the third example would not be sufficient to return the correct value of maximum.