Problem A. Doubt VS Lie

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Doubt VS Lie is a kind of poker game. Initially, there are 4 players denoted by $0, 1, 2, 3$. For player x define $S(x)$ as the set of cards he/she holds currently. At the very beginning $|S(x)| = 13, 0 \le x \le 3$, for every $0 \le x, y \le 3, x \ne y, S(x) \cap S(y) = \emptyset$. $\bigcup S(x)$ is exactly a deck without two Jokers. As you will know later, the suit of cards doesn't matter in this game. So, you can assume $\bigcup S(x)$ contains exactly 4 Ace, 4 Two, 4 Three, ..., 4 Queen and 4 King. For convenience, in the input/output and the following statement, we denote Ace by "1", Jack by "11", Queen by "12", King by "13" and others by their number (2 to 10).

The process of the game is as follows:

1. First, player 0 chooses an $x(1 \le x \le 13)$ and arbitrarily $y(1 \le y \le |S(0)|)$ cards in his/her hand, playing these y cards with the back facing up on the desk and then state "These are y pieces of card x".
2. When player p has done his/her operation, it comes to the turn of player q ($q = (p + 1) \bmod 4$).
   (a) If player p played cards and stated "These are y pieces of card x", the player q can choose:
      i. To play arbitrarily $z(1 \le z \le |S((p + 1) \bmod 4)|)$ cards in his/her hand with the back facing upon the desk and then state "These are z pieces of card x". Note that x should equal to what player p stated.
      ii. Not to play cards and call into question. Then he will flip over the card the player p has played and check if his/her statement is true. If it's true, the doubt is failed and player q should take all the cards on the desk into his/her hand. Otherwise, the doubt is successful and player p should take all the cards on the desk into his/her hand.
   (b) If player p called a question, whether the doubt was successful, player q choose an $x(1 \le x \le 13)$ and arbitrarily $y(1 \le y \le |S(q)|)$ cards in his/her hand, playing these y cards with the back facing up on the desk and then state "These are y pieces of card x" like player 0 did initially. Note that at this time x can be chosen arbitrarily.
3. When a player chooses to call a question, after someone takes all the cards on the desk, if somebody has no cards in his/her hand, then we say that player wins the game.

For simplification, this time you need to write a program only to simulate the game process.

Input

Firstly 4 lines, the i-th line contains 13 numbers, indicating the cards player $(i - 1)$ has in hand.

Then one line contains an integer m ($2 \le m \le 20$), denoting the turns of the game. The next m lines, each of which is one of the following formats (without quotes):

1. "S x y a_1 a_2 ... a_y", meaning that the current player plays card $a_1, a_2, \cdots, a_y$ and stated "These are y pieces of card x".
2. "! y a_1 a_2 ... a_y", meaning that the current player chooses not to call a question and played y cards $a_1, a_2, \cdots, a_y$ and states "These are y pieces of card x" where x remains the same.
3. "?", meaning that the current player chooses to call a question.

Tests guarantees that all the operations are legal. And in the process, nobody will win the game.

Output

Four lines, the i-th line contains the cards of player $(i-1)$ after m turns in non-decreasing order.

Examples

standard input	standard output
8 12 12 9 3 10 13 11 12 4 10 2 1 2 10 13 9 9 3 12 4 6 13 3 11 13 11 1 10 5 7 4 5 6 7 7 5 6 9 4 1 11 2 1 8 8 3 2 6 5 8 7 5 S 1 2 2 3 ! 1 2 ! 1 1 ! 2 1 1 ?	1 1 1 1 2 2 3 4 8 9 10 10 11 12 12 12 13 3 3 4 6 9 9 10 11 12 13 13 13 4 5 5 5 6 6 7 7 7 9 10 11 2 2 3 4 5 6 7 8 8 8 11
8 12 12 9 3 10 13 11 12 4 10 2 1 2 10 13 9 9 3 12 4 6 13 3 11 13 11 1 10 5 7 4 5 6 7 7 5 6 9 4 1 11 2 1 8 8 3 2 6 5 8 7 6 S 1 2 2 3 ! 1 2 ! 1 1 ! 2 1 1 ! 1 1 ?	4 8 9 10 10 11 12 12 12 13 1 1 1 1 2 2 3 3 3 4 6 9 9 10 11 12 13 13 13 4 5 5 5 6 6 7 7 7 9 10 11 2 2 3 4 5 6 7 8 8 8 11
8 12 12 9 3 10 13 11 12 4 10 2 1 2 10 13 9 9 3 12 4 6 13 3 11 13 11 1 10 5 7 4 5 6 7 7 5 6 9 4 1 11 2 1 8 8 3 2 6 5 8 7 8 S 3 1 3 ! 2 2 4 ! 1 4 ? S 4 1 2 ! 1 3 ! 1 4 ?	1 4 8 9 10 10 11 12 12 12 13 3 6 9 9 10 11 12 13 13 13 1 2 3 4 5 5 5 6 6 7 7 7 9 10 11 1 1 2 2 2 3 3 4 4 5 6 7 8 8 8 11

Problem B. Degree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

3 is a mysterious number with a lot of anecdotes. It means the empress in TAROT. Pythagoras used it to describe the male. And Gabe can't count to 3.

So there's an interesting challenge about 3 for you. There are a lot of teams now participating in CCPC. However, the huge number of teams makes it hard to connect all computers to the local network considering the difficulty in arranging network cables. To find a solution, you decide to pick up some of the computers and wish to get some spark in a simplified situation. The connection of computers can be regarded as an undirected graph. To keep the topology of the network graph unchanged, you decide to choose some computers and their degrees modulo 3 won't change in your newly chosen connection plan. Nobody knows why you choose the number 3 but we all know 3 is mysterious. What's more, you must hurry up now. If you can't finish it in time, you will get how-to-comment in Zhihu of course.

Formally, you are given a simple undirected graph with n vertices and m edges. Now you need to pick up some of the vertices(at least one and not n) to build a new graph(induced subgraph). Each vertex's degree in the new graph must have the same remainder divided by three as it's in the original graph.

In the mathematical field of graph theory, an induced subgraph of a graph is another graph, formed from a subset of the vertices of the graph and all of the edges (from the original graph) connecting pairs of vertices in that subset.

Input

The first line contains two integers $n, m (1 \le n \le 5 \times 10^5, 0 \le m \le 10^6)$.

The next m lines, each line contains two integers $x, y (1 \le x, y \le n, x \ne y)$ denoting an edge between x and y.

Output

If there's no plan, output **No**;

otherwise, output **Yes**,

then output a single integer k $(1 \le k \le n-1)$ in the second line meaning the number of vertices in the new graph, and the third line contains k integers indicating the index of vertices in the new graph.

Note that:

1. You mustn't pick up the whole graph.
2. You must guarantee the vertices are different in the output.
3. If there are multiple answers, print any of them. And you can print the vertices in any order.

Examples

standard input	standard output
3 3 1 2 2 3 3 1	No
6 6 1 2 2 3 3 1 4 5 5 6 6 4	Yes 3 1 2 3
6 9 1 2 2 3 3 4 4 1 1 5 1 6 1 3 3 5 3 6	Yes 3 3 2 1

Problem C. Guess

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

Alice and Bob are playing a game. Each of them has a card on his/her forehead with a positive integer written on it(Alice has the integer a and Bob has the integer b).

Alice and Bob can only see their opponent's number, but both of them know that the two numbers a, b are positive integers and one is twice as large as the other.

Each round, Alice and Bob try to guess their own number in turn, Alice first. (Each round ends when both Alice and Bob finish their tries.)

Both of them are smart enough and they won't make mistakes. They will speak out the answer if and only if they exactly know it, otherwise, they will say "I don't know"instead.

Given the two numbers on Alice's and Bob's foreheads, find out which round one of them can speak out the integer on his/her forehead.

Input

This problem contains multiple test cases.

The first line contains an integer T indicating the number of test cases ($1 \le T \le 10^5$).

For each test case, the only line contains two integers a, b ($1 \le a, b \le 10^{18}$).

It is guaranteed that $a = 2b$ or $b = 2a$ always holds.

Output

For each test case, output two integers r, p indicating at the r-th round, p can speak out the answer. $p = 0$ stands for Alice and $p = 1$ stands for Bob.

Example

standard input	standard output
3 6 3 1 2 49999999999999999 99999999999999998	1 0 1 1 1 1

Note

For the first sample, Alice can see the number on Bob's forehead, which is an odd number 3. Then she can find out that the number on her forehead must be 6 because there is no integer x such that $3 = 2x$.

Please notice that: The range of a, b is just for you while the two players don't know it. That is, when $a = 5 \times 10^{17}, b = 10^{18}$, Alice can't speak out her number immediately even if $a = 2 \times 10^{18}$ is out of input range.

Problem D. Pointer Sequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Give you a pointer sequence of length n, which is initialized as follows:

```
using integer = long long;
integer *A[MAXN]; //MAXN is a constant larger than n
for (int i = 1; i <= n; ++i)
    A[i] = new integer(0);
```

You need to maintain five types of operations, as shown below:

- **1 L1 R1 L2 R2** $(R1 - L1 = R2 - L2, [L1, R1]$ and $[L2, R2]$ are disjoint intervals falling in $[1, n])$
- **2 L R k b** $(1 \le L \le R \le n$ and $0 \le k, b \le 10^4)$
- **3 L R v** $(0 \le L \le R \le 10^9$ and $0 \le v \le 10^9)$
- **4 p v** $(1 \le p \le n$ and $0 \le v \le 10^9)$
- **5 p** $(1 \le p \le n)$

For each type of the operations, you need to efficiently handle the corresponding procedure:

Operation 1

```
for (int i = 0; i <= R1 - L1; ++i) //copy A[L1, R1] to A[L2, R2]
    A[L2 + i] = A[L1 + i];
```

Operation 2

```
for (integer i = 0; i <= R - L; ++i) //assign an arithmetic sequence to A[L, R]
    A[L + i] = new integer(k * i + b);
```

Operation 3

```
for (int i = 1; i <= n; ++i) //change the value in [L, R] to v
    if (*A[i] >= L && *A[i] <= R)
        *A[i] = v;
```

Operation 4

```
*A[p] = v; //single point modification
```

Operation 5

```
cout << *A[p] << endl; //Output the value of A[pos]
```

Input

The first line of the input contains two integers $n(1 \le n \le 10^6)$ and $m(1 \le m \le 10^5)$, indicating the length of the sequence and the number of operations. Then each of the next m lines contains a query in the form described above.

Output

For all operations of type 5, output an integer indicating the answer.

Example

standard input	standard output
5 10 5 1 2 1 5 1 1 1 1 2 3 4 4 1 8 5 3 3 7 8 2 3 2 5 6 5 1 5 2 5 5	0 8 6 6 6

Problem E. Substring Match

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given two strings s and t, s is a string containing only lowercase letters, and t is a pattern containing lowercase and uppercase letters.

Defines that a string can be matched by a pattern:

- A lowercase letter in the pattern matches a corresponding lowercase letter.
- An uppercase letter in the pattern matches any number (including zero) of the corresponding lowercase letters.

For example, abb can be matched by aB; aaa can be matched by A; ac can be matched by aBc; ABC, $aABCc$, aab can not be matched by aB; aa can not be matched a.

Now you need to answer the length of the longest substring of s that can be matched by t.

Input

This problem contains multiple test cases.

The first line contains an integer $T(1 \le T \le 100)$ indicating the number of test cases.

For each test case, the first line contains a string $s(1 \le |s| \le 10^5)$, the second line contains a string $t(1 \le |t| \le 10^5)$.

It's guaranteed that $\sum |s| \le 10^5$, $\sum |t| \le 10^5$, and **total of uppercase letters of t no more than** 200 **for all test cases**.

Output

For each test case, output the length of the longest substring of s that can be matched by t in a line.

Example

standard input	standard output
3 aaa ABa aaa aAaBa aaa aa	3 3 2

Problem F. Xor Circle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

CCPC (China Collegiate Programming Contest) will be held in city A. City A can be considered as an infinity 2-dimensional plane. n contestants will take part in the competition. These n people are numbered from 1 to n, each of them has an initial position in city A and a value a_i.

The organizer needs to choose a place for the competition site. For each person i, he/she will be happy if and only if the distance from the competition site to his/her initial position is not greater than d. Note that the coordinate of the site can be any real number.

To determine whether the competition site is good or not, the organizer defines $f(P)$ as the maximum xor sum of **some** of the happy people's value if point P is considered as the competition site.

Let $S(P)$ denotes all person such that distance between P and his/her location is not greater than d, $g(T)$ denotes the exclusive OR of all a_i such that person i is in T (**Specially**, $g(\varnothing) = 0$), $f(P) = \max_{T \subseteq S(P)} g(T)$. A point P is considered good if and only if $f(P) \geq k$.

You need to find out the minimum non-negative real number d, such that there are at least one location is good.

Note that the distance between point $A(x_A, y_A)$ and $B(x_B, y_B)$ is $\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$.

Input

The first line contains two integers $n(1 \leq n \leq 1000), k(0 \leq k < 2^{20})$ - the number of the competitors and the minimum value of a good location.

For the next n lines, each line contains three integers $x_i, y_i, a_i(x_i, y_i \leq |10^4|, 0 \leq a_i < 2^{20})$ - the initial position and the value of the i-th person.

Output

A single non-negative real number d - the minimum distance such that there at least one location's value is not smaller than k.

Your answer will be considered correct if and only if the absolute or relative error is not greater than 10^{-6}.

If the answer does not exist, please output `-1`.

Examples

standard input	standard output
4 39 -1 -1 5 2 3 37 -1 2 2 -1 -1 14	1.5811388301
4 16 1 1 4 5 1 4 1 9 1 9 8 10	-1
2 0 11 45 14 191 98 10	0.0000000000

Problem G. Name the Puppy

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Yuan bought a cute puppy for Chinese Valentine's Day. But before telling his lover, he needs to name the puppy.

Let $\overline{s}$ denote the reverse of string s. Given a string s of length n, we call s' of length $n'(n' < n)$ an **anti-border** of s if s' is a prefix of s and $\overline{s'}$ is also a suffix of s.

His lover loves **anti-border** so much that Yuan decides to choose a name which has the longest **anti-border** for the puppy. However, Yuan can only choose a name that he can spell.

Formally speaking, the set of words Yuan can spell is $\{s_1, s_2, s_3, \cdots, s_n\}$. The set of all possible names S is defined as follows:

- $s_i \in S (1 \le i \le n)$.
- If $x, y \in S$, then $x + y \in S$. The $+$ here means simple string concatenation.

Let $f(s)$ denote the length of the longest **anti-border** of s. Please help Yuan calculate $\max\limits_{x \in S}\{f(x)\}$ or tell him the result is infinity.

Input

The first line contains an integer n, denoting the number of strings.

The $(i+1)$-th line $(1 \le i \le n)$ contains a string s_i.

It is guaranteed that $1 \le n \le 5000, 1 \le \sum\limits_{i=1}^{n} |s_i| \le 5000$ and words only consist of lowercase letters.

Output

Output `INF` if the result is infinity, output an integer indicating the result otherwise.

Examples

standard input	standard output
4 ab cbba bccd eddc	6
3 abcdefg gfe dcba	INF

Note

In the first test case, $f(x) = 6$ when $x = s_1 + s_3 + s_4 + s_2$. It can be proved that there is no x such that $x \in S, f(x) > 6$. So the result is 6.

Problem H. Mutiple Set

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Peter is obsessed with mathematics. He often works on interesting mathematical problems. Once he became interested in the problem of multiples. Let $\mathcal{S}_{[L,R]}(x)$ be the set of multiples of x in the interval $[L, R]$. Formally, $\mathcal{S}_{[L,R]}(x) = \{d | L \leq d \leq R, x|d\}$.

He defines the weight of a set as the sum of all elements of the set. Then he sums up the weight of all non-empty subsets of $\mathcal{S}_{[L,R]}(x)$ and obtains the result K.

Unfortunately, Peter is forgetful. After he has accomplished the calculation tasks, he forgets what the initial x is. However, he has written down the interval $[L, R]$ and the result K. Now he wants you, a smart XCPCer, to tell him all the possible x.

Input

This problem contains multiple test cases.

In the first line, one integer $T(1 \leq T \leq 100)$ represents the number of calculation tasks Peter has accomplished.

In the following T lines, each line contains three integers $L, R(1 \leq L \leq R \leq 10^{12})$ and $K(1 \leq K \leq 10^{14})$, which represent the interval and the result respectively.

Output

For each task, if there is no legal x, output `No Solution`. If the number of legal x is over 10^5, output `Too Many!`. Otherwise, output the number of legal x in the first line, then output all legal x in ascending order in the next line, separated by spaces.

Example

standard input	standard output
10 25 37 25 10 38 28 53 112 82 343 1839 1373 2451576 8343226 60034488 3402021 5387344 3855560 1332880393 1809278150 6322710186 79896573 1625195292 12360010488 217028134566 876902973439 657118365564 693979750091 779344640488 2981730890110	1 25 1 28 2 41 82 1 1373 1 2501437 2 1927780 3855560 2 243181161 351261677 1 515000437 1 657118365564 1 42596155573

Problem I. Transport Plan

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

There are n cities and m transport plans. For each integer i in $[1, n-1]$, there is a bidirectional road between the i-th city and the $(i+1)$-th city with 1km length. The i-th transport plan is to transport v_i ton goods from the a_i-th city to the b_i-th city.

It costs 1 yuan to transport 1 ton of goods for 1km. You can choose two cities x, y and build a bidirectional portal between them. Goods can be transported between these two cities at no cost.

What is the minimum cost to accomplish all the m transport plans?

Input

The first line contains two integers n, m ($2 \le n \le 3000, 1 \le m \le 2 \times 10^5$).

For the next m lines, the i-th line contains three numbers a_i, b_i, v_i ($1 \le a_i, b_i \le n, 1 \le v_i \le 10^6$).

Output

One integer, denoting the minimum cost.

Example

standard input	standard output
4 3 1 3 2 4 2 3 1 4 4	5

Problem J. Count Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Alice loves doing research on permutations. For a n-permutation $P = (p_1, p_2, ..., p_n)$, she defines $w(P) = |S|$, where $S = \{p_i - p_{(i \bmod n)+1} | 1 \le i \le n\}$. For example, if $P = (2, 4, 5, 3, 1)$, then $S = \{2, -1, -2\}, w(P) = |S| = 3$.

Alice hopes S to be as simple as possible, so she wants to find all n-permutation P such that $w(P)$ is minimal (She defines these permutations as "good permutation"). However, the number of good permutations may be too large. As a result, she writes down three numbers m, s, t and wants you to find out the number of good permutations $P = (p_1, p_2, ..., p_n)$ that $p_1 = s$ and $p_m = t$.

Input

One line that contains 4 integers $n, m, s, t, (3 \le n \le 10^{12}, 2 \le m \le n, 1 \le s, t \le n, s \ne t)$.

Output

One line, an integer, denoting the answer.

Examples

standard input	standard output
3 2 1 3	1
4 3 1 3	2

Note

A n-permutation is an integer sequence whose length is n and each element of $\{1, 2, ..., n\}$ appears exactly once.

Problem K. Roulette

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

This is an interactive problem. Please be sure to use the stream flushing operation after each query's output in order not to leave part of your output in some buffer. For example, you've got to use the `fflush(stdout)` in C/C++, `sys.stdout.flush()` in Python(do not forget `import sys`) and `System.out.flush()` in Java.

Roulette is a kind of gambling. For simplification, we assume its rules and steps as follows:

1. The whole gambling process composes of many turns.
2. In each turn:
   - You can choose an x and pay x yuan as the wager. x could be any positive integer in range $[1, 10^{700})$.
   - The maker will turn the roulette which has only two results indicated by two colors: black and white.
   - If the result corresponds to the white color, you will get $2x$ yuan from the maker, which means you gain x yuan in this turn. Otherwise, the maker will devour the x yuan you have paid, which means you lose x yuan in this turn.
3. After each turn, you can choose to continue gambling or just stop it.
4. The probability of two results are equal. That means both of them are 0.5.

Now a profiteer set a roulette game and lure people taking part in it. In order to defraud more money, he can set the probability of the roulette's result. More precisely, he can set the probability you win to p ($0 < p < 1$, accordingly, the probability you lose is $1 - p$) in advance and keep that probability until the end of game.

Unfortunately, little A is rich enough but failed to bear the endure and has lost y yuan due to gambling. Now he asks you, a smart ACMer, to help him win back his money. You don't want to use any money of yourself for gambling. So, he will lend you 10^{700} yuan–a large sum of money, hoping you to take part in that game to win back his money (that means, initially you have 10^{700} yuan, you should have at least $10^{700} + y$ yuan sometime and at any time you should have a positive amount of money all the time). You also don't want to waste time, so you hope to gamble for no more than 5000 turns.

Input

The first line includes three integers: $y(1 \le y \le 1145141919810), q(1 \le q \le 100), seed(1 \le seed \le 10^{18})$, indicates that little A lost y yuan and the profiteer set the probability p(you win) to $\frac{q}{100}$, $seed$ is used by the interactor to produce your win-lose sequence. **Your program needs to do nothing on it but only read it.** Your program will receive "0","1" or "2"(without quotes) as the result of the current turn where "0" means that you lose in this turn, "1" means you win in this turn but still haven't win back enough money and "2" means you win in this turn and have win back enough money(under this condition, to prevent unnecessary errors, please let your program terminate at once to get "Accepted". Otherwise, we have nothing to do but wish you good luck). Your program should read these results and make decisions accordingly.

Output

Several lines, each of which has the format x(without quotes, x is an positive integer in range $[1, 10^{700})$ **without leading zeros**), means that you continue the game and pay x yuan as the wager.

Interaction Protocol

The interactor will read your output. Then it will use *seed* to randomly produce a real number p' in range $[0, 1)$ and compare it with p. If $p' < p$, it will send "0", otherwise it will send "1" or "2" (See the input section). **You may assume that your win-lose sequence is unique and determined by** *seed***.**

Under these (including but not limited to) conditions you may get result "Wrong answer", "Time limit exceed" or other incorrect results. Please check your code carefully before submitting.

1. Wrong output format.
2. Having used excessive amount of money.
3. Failed to win back enough money in 5000 turns.

Examples

standard input	standard output
1145141 91 9810 1 0 2	114514 1145141 11451419
100 100 2333 2	200

Problem L. Tree Division

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	512 megabytes

Given a tree with n vertices. For an integer k, you should delete no less than k edges in the tree, so that the smallest connected component is as large as possible.

Please calculate the maximum size of the smallest connected component for each $k(0 \le k \le n-1)$.

Input

The first line contains one integer n $(1 \le n \le 10^5)$.

The next $n-1$ lines each contains two integers u and v $(1 \le u, v \le n, u \ne v)$, representing an edge (u, v).

Output

Output n lines, the i-th line contains a single integer indicating the answer when $k = i - 1$.

Example

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

Problem M. Count Permutation 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given an array $p_1, p_2, \cdots, p_n$ and an integer k. It is guaranteed that the array is a permutation initially ($1 \le p_i \le n$, and for each $1 \le i < j \le n$,$p_i \ne p_j$). You can do the following operation any number of times:

- Choose an integer j ($1 \le j \le n-k+1$), define $m = \max(p_j, p_{j+1}, \cdots, p_{j+k-1})$, and then let $p_i = m$ for all integers $i \in [j, j+k-1]$ simultaneously.

Find the minimum number of operations to make all the elements in the array p equal to n.

Input

The first line contains two integers n, k ($2 \le n \le 1 \times 10^5, 2 \le k \le n$), denoting the length of the array and the parameter in one operation.

The second line contains n integers $p_1, p_2, \cdots, p_n$ ($1 \le p_i \le n$), denoting the array p.

Output

A single line contains an integer m, denoting the minimum number of operations.

Example

standard input	standard output
5 3 1 5 4 2 3	2