Codeforces Round #275 (Div. 1)

A. Diverse Permutation

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

Permutation p is an ordered set of integers $p_1, p_2, ..., p_n$, consisting of n distinct positive integers not larger than n. We'll denote as n the length of permutation $p_1, p_2, ..., p_n$.

Your task is to find such permutation p of length n, that the group of numbers $|p_1 - p_2|, |p_2 - p_3|, ..., |p_{n-1} - p_n|$ has exactly k distinct elements.

Input

The single line of the input contains two space-separated positive integers n, k $(1 \leq k < n \leq 10^5)$.

Output

Print n integers forming the permutation. If there are multiple answers, print any of them.

Sample test(s)

input
3 2
output
1 3 2

input
3 1
output
1 2 3

input
5 2
output
1 3 2 4 5

Note

By $|x|$ we denote the absolute value of number x.

B. Interesting Array

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

We'll call an array of n non-negative integers $a[1], a[2], ..., a[n]$ *interesting*, if it meets m constraints. The i-th of the m constraints consists of three integers l_i, r_i, q_i ($1 \le l_i \le r_i \le n$) meaning that value $a[l_i] \ \& \ a[l_i + 1] \ \& \ \ldots \ \& \ a[r_i]$ should be equal to q_i.

Your task is to find any *interesting* array of n elements or state that such array doesn't exist.

Expression $x\&y$ means the bitwise AND of numbers x and y. In programming languages C++, Java and Python this operation is represented as "&", in Pascal — as "and".

Input

The first line contains two integers n, m ($1 \le n \le 10^5$, $1 \le m \le 10^5$) — the number of elements in the array and the number of limits.

Each of the next m lines contains three integers l_i, r_i, q_i ($1 \le l_i \le r_i \le n$, $0 \le q_i < 2^{30}$) describing the i-th limit.

Output

If the *interesting* array exists, in the first line print "YES" (without the quotes) and in the second line print n integers $a[1], a[2], ..., a[n]$ ($0 \le a[i] < 2^{30}$) decribing the *interesting* array. If there are multiple answers, print any of them.

If the *interesting* array doesn't exist, print "NO" (without the quotes) in the single line.

Sample test(s)

input
3 1 1 3 3
output
YES 3 3 3

input
3 2 1 3 3 1 3 2
output
NO

C. Game with Strings

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

You play the game with your friend. The description of this game is listed below.

Your friend creates n distinct strings of the same length m and tells you all the strings. Then he randomly chooses one of them. He chooses strings equiprobably, i.e. the probability of choosing each of the n strings equals $\frac{1}{n}$. You want to guess which string was chosen by your friend.

In order to guess what string your friend has chosen, you are allowed to ask him questions. Each question has the following form: «What character stands on position pos in the string you have chosen?» A string is considered guessed when the answers to the given questions uniquely identify the string. After the string is guessed, you stop asking questions.

You do not have a particular strategy, so as each question you equiprobably ask about a position that hasn't been yet mentioned. Your task is to determine the expected number of questions needed to guess the string chosen by your friend.

Input

The first line contains a single integer n ($1 \le n \le 50$) — the number of strings your friend came up with.

The next n lines contain the strings that your friend has created. It is guaranteed that all the strings are distinct and only consist of large and small English letters. Besides, the lengths of all strings are the same and are between 1 to 20 inclusive.

Output

Print the single number — the expected value. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-9}.

Sample test(s)

input
2 aab aac
output
2.000000000000000

input
3 aaA aBa Caa
output
1.666666666666667

input
3 aca vac wqq
output
1.000000000000000

Note

In the first sample the strings only differ in the character in the third position. So only the following situations are possible:

- you guess the string in one question. The event's probability is $\frac{1}{3}$;
- you guess the string in two questions. The event's probability is $\frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}$ (as in this case the first question should ask about the position that is other than the third one);
- you guess the string in three questions. The event's probability is $\frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{1} = \frac{1}{3}$;

Thus, the expected value is equal to $\frac{1+2+3}{3} = 2$

In the second sample we need at most two questions as any pair of questions uniquely identifies the string. So the expected number of questions is $\frac{1}{3} + 2 \cdot \frac{2}{3} = \frac{5}{3}$.

In the third sample whatever position we ask about in the first question, we immediately identify the string.

D. Random Function and Tree

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

You have a rooted tree consisting of n vertices. Let's number them with integers from 1 to n inclusive. The root of the tree is the vertex 1. For each $i > 1$ direct parent of the vertex i is p_i. We say that vertex i is child for its direct parent p_i.

You have initially painted all the vertices with red color. You like to repaint some vertices of the tree. To perform painting you use the function `paint` that you call with the root of the tree as an argument. Here is the pseudocode of this function:

```
count = 0 // global integer variable

rnd() { // this function is used in paint code
   return 0 or 1 equiprobably
}

paint(s) {
   if (count is even) then paint s with white color
   else paint s with black color

   count = count + 1

   if rnd() = 1 then children = [array of vertex s children in ascending order of their numbers]
   else children = [array of vertex s children in descending order of their numbers]

   for child in children { // iterating over children array
      if rnd() = 1 then paint(child) // calling paint recursively
   }
}
```

As a result of this function, some vertices may change their colors to white or black and some of them may remain red.

Your task is to determine the number of distinct possible colorings of the vertices of the tree. We will assume that the coloring is possible if there is a nonzero probability to get this coloring with a single call of $paint(1)$. We assume that the colorings are different if there is a pair of vertices that are painted with different colors in these colorings. Since the required number may be very large, find its remainder of division by 1000000007 $(10^9 + 7)$.

Input

The first line contains a single integer n $(2 \le n \le 10^5)$ — the number of vertexes in the tree.

The second line contains n - 1 integers $p_2, p_3, ..., p_n$ $(1 \le p_i < i)$. Number p_i is the parent of vertex i.

Output

Print a single integer — the answer to the problem modulo 1000000007 $(10^9 + 7)$

Sample test(s)

input
4 1 2 1

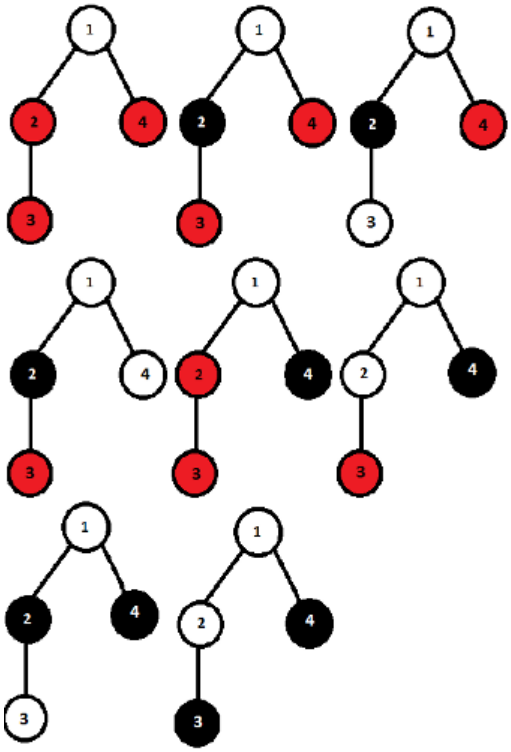
output
8

input
3 1 1

output
5

Note

All possible coloring patterns of the first sample are given below.


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

E. ELCA

time limit per test: 8 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a root tree containing n vertexes. Let's number the tree vertexes with integers from 1 to n. The tree root is in the vertex 1.

Each vertex (except fot the tree root) v has a direct ancestor p_v. Also each vertex v has its integer value s_v.

Your task is to perform following queries:

- **P** v u ($u \neq v$). If u isn't in subtree of v, you must perform the assignment $p_v = u$. Otherwise you must perform assignment $p_u = v$. Note that after this query the graph continues to be a tree consisting of n vertexes.
- **V** v t. Perform assignment $s_v = t$.

Your task is following. Before starting performing queries and after each query you have to calculate expected value written on the lowest common ancestor of two equiprobably selected vertices i and j. Here lowest common ancestor of i and j is the deepest vertex that lies on the both of the path from the root to vertex i and the path from the root to vertex j. Please note that the vertices i and j can be the same (in this case their lowest common ancestor coincides with them).

Input

The first line of the input contains integer n ($2 \le n \le 5 \cdot 10^4$) — the number of the tree vertexes.

The second line contains n - 1 integer $p_2, p_3, ..., p_n$ ($1 \le p_i \le n$) — the description of the tree edges. It is guaranteed that those numbers form a tree.

The third line contains n integers — $s_1, s_2, ... s_n$ ($0 \le s_i \le 10^6$) — the values written on each vertex of the tree.

The next line contains integer q ($1 \le q \le 5 \cdot 10^4$) — the number of queries. Each of the following q lines contains the description of the query in the format described in the statement. It is guaranteed that query arguments u and v lie between 1 and n. It is guaranteed that argument t in the queries of type **V** meets limits $0 \le t \le 10^6$.

Output

Print $q + 1$ number — the corresponding expected values. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-9}.

Sample test(s)

input
5 1 2 2 1 1 2 3 4 5 5 P 3 4 P 4 5 V 2 3 P 5 2 P 1 4
output
1.640000000 1.800000000 2.280000000 2.320000000 2.800000000 1.840000000

Note

Note that in the query **P** v u if u lies in subtree of v you must perform assignment $p_u = v$. An example of such case is the last query in the sample.