A. Increasing by Modulo

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

Toad Zitz has an array of integers, each integer is between 0 and $m-1$ inclusive. The integers are $a_1, a_2, \ldots, a_n$.

In one operation Zitz can choose an integer k and k indices $i_1, i_2, \ldots, i_k$ such that $1 \leq i_1 < i_2 < \ldots < i_k \leq n$. He should then change a_{i_j} to $((a_{i_j}+1) \bmod m)$ for each chosen integer i_j. The integer m is fixed for all operations and indices.

Here $x \bmod y$ denotes the remainder of the division of x by y.

Zitz wants to make his array non-decreasing with the minimum number of such operations. Find this minimum number of operations.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 300\,000$) — the number of integers in the array and the parameter m.

The next line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i < m$) — the given array.

Output

Output one integer: the minimum number of described operations Zitz needs to make his array non-decreasing. If no operations required, print 0.

It is easy to see that with enough operations Zitz can always make his array non-decreasing.

Examples

input

```
5 3
0 0 0 1 2
```

output

```
0
```

input

```
5 7
0 6 1 3 2
```

output

```
1
```

Note

In the first example, the array is already non-decreasing, so the answer is 0.

In the second example, you can choose $k=2$, $i_1 = 2$, $i_2 = 5$, the array becomes $[0,0,1,3,3]$. It is non-decreasing, so the answer is 1.

B. Good Triple

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Toad Rash has a binary string s. A binary string consists only of zeros and ones.

Let n be the length of s.

Rash needs to find the number of such pairs of integers l, r that $1 \leq l \leq r \leq n$ and there is at least one pair of integers x, k such that $1 \leq x, k \leq n$, $l \leq x < x + 2k \leq r$, and $s_x = s_{x+k} = s_{x+2k}$.

Find this number of pairs for Rash.

Input

The first line contains the string s ($1 \leq |s| \leq 300\,000$), consisting of zeros and ones.

Output

Output one integer: the number of such pairs of integers l, r that $1 \le l \le r \le n$ and there is at least one pair of integers x, k such that $1 \le x, k \le n$, $l \le x < x + 2k \le r$, and $s_x = s_{x+k} = s_{x+2k}$.

Examples

input
```
010101
```
output
```
3
```

input
```
11001100
```
output
```
0
```

Note

In the first example, there are three l, r pairs we need to count: 1, 6; 2, 6; and 1, 5.

In the second example, there are no values x, k for the initial string, so the answer is 0.

C. And Reachability

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

Toad Pimple has an array of integers $a_1, a_2, \ldots, a_n$.

We say that y is reachable from x if $x < y$ and there exists an integer array p such that $x = p_1 < p_2 < \ldots < p_k = y$, and $a_{p_i} \,\&\, a_{p_{i+1}} > 0$ for all integers i such that $1 \le i < k$.

Here $\&$ denotes the bitwise AND operation.

You are given q pairs of indices, check reachability for each of them.

Input

The first line contains two integers n and q ($2 \le n \le 300\,000$, $1 \le q \le 300\,000$) — the number of integers in the array and the number of queries you need to answer.

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 300\,000$) — the given array.

The next q lines contain two integers each. The i-th of them contains two space-separated integers x_i and y_i ($1 \le x_i < y_i \le n$). You need to check if y_i is reachable from x_i.

Output

Output q lines. In the i-th of them print "Shi" if y_i is reachable from x_i, otherwise, print "Fou".

Example

input
```
5 3
1 3 0 2 1
1 3
2 4
1 4
```
output
```
Fou
Shi
Shi
```

Note

In the first example, $a_3 = 0$. You can't reach it, because AND with it is always zero. $a_2 \,\&\, a_4 > 0$, so 4 is reachable from 2, and to go from 1 to 4 you can use $p = [1, 2, 4]$.

D. Anagram Paths

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Toad Ilya has a rooted binary tree with vertex 1 being the root. A tree is a connected graph without cycles. A tree is rooted if one vertex is selected and called the root. A vertex u is a child of a vertex v if u and v are connected by an edge and v is closer to the root than u. A leaf is a non-root vertex that has no children.

In the tree Ilya has each vertex has **at most two** children, and each edge has some character written on it. The character can be a lowercase English letter or the question mark '?'.

Ilya will q times update the tree a bit. Each update will replace exactly one character on some edge. After each update Ilya needs to find if the tree is *anagrammable* and if yes, find its *anagramnity* for each letter. Well, that's difficult to explain, but we'll try.

To start with, a string a is an *anagram* of a string b if it is possible to rearrange letters in a (without changing the letters itself) so that it becomes b. For example, the string "fortyfive" is an anagram of the string "overfifty", but the string "aabb" is not an anagram of the string "bbba".

Consider a path from the root of the tree to a leaf. The characters on the edges on this path form a string, we say that this string is associated with this leaf. The tree is *anagrammable* if and only if it is possible to replace each question mark with a lowercase English letter so that for all pair of leaves the associated strings for these leaves are anagrams of each other.

If the tree is *anagrammable*, then its *anagramnity* for the letter c is the maximum possible number of letters c in a string associated with some leaf in a valid replacement of all question marks.

Please after each update find if the tree is *anagrammable* and if yes, find the $\sum f(c) \cdot ind(c)$ for all letters c, where $f(c)$ is the *anagramnity* for the letter c, and $ind(x)$ is the index of this letter in the alphabet (ind("a")$= 1$, ind("b")$= 2$, ..., ind("z")$= 26$).

Input

The first line of input contains two integers n and q ($2 \le n \le 150\,000$, $1 \le q \le 150\,000$) — the number of vertices in the tree and the number of queries.

The next $n-1$ lines describe the initial tree. The i-th of them contains an integer p_i and a character c_i ($1 \le p_i \le i$, c_i is a lowercase English letter or the question mark '?') describing an edge between vertices p_i and $i+1$ with character c_i written on it.

The root of this tree is the vertex 1, and each vertex has at most two children.

The next q lines describe the queries. The i-th of them contains two integers v and c ($2 \le v \le n$, c is a lowercase English letter or the question mark '?'), meaning that updated character on the edge between p_{v-1} to v is c. The updated character can be the same as was written before.

Output

Output q lines. In the i-th of them print "Fou" if the tree is **not** *anagrammable* after the first i updates.

Otherwise output "Shi" and the $\sum f(c) \cdot ind(c)$ for all letters c.

Examples

input

```
3 4
1 ?
1 ?
2 ?
2 a
3 b
2 b
```

output

```
Shi 351
Shi 1
Fou
Shi 2
```

input

```
5 2
1 ?
1 ?
2 ?
3 ?
4 a
5 b
```

output

```
Shi 352
Shi 3
```

Note

In the first example after the first query, for each character, you can set all edges equal to that character, and you will get 1 such character on each path, so the answer is $1 \cdot (1 + 2 + \ldots + 26) = 351$.

In the first example after the second query, you know that all paths should be an anagram of "a", so all paths should be "a", so the answer is $1 \cdot 1 = 1$.

In the first example after the third query, you have two paths with strings "a" and "b", but these strings are not anagrams, so the answer is "Fou".

In the first example after the fourth query, you know that all paths should be "b", so the answer is $1 \cdot 2 = 2$.

In the second example after the first query, you know that f('a')$= 2$ and $f(c) = 1$ for all other characters, so the answer is

$1 \cdot (2 + 3 + \ldots + 26) + 2 = 352$.

In the second example after the second query, you know that each path should contain one 'a' and one 'b', so the answer is $1 \cdot 1 + 1 \cdot 2 = 3$.

E. Xor Permutations

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

Toad Mikhail has an array of 2^k integers $a_1, a_2, \ldots, a_{2^k}$.

Find two permutations p and q of integers $0, 1, \ldots, 2^k - 1$, such that a_i is equal to $p_i \oplus q_i$ for all possible i, or determine there are no such permutations. Here $\oplus$ denotes the bitwise XOR operation.

Input

The first line contains one integer k ($2 \le k \le 12$), denoting that the size of the array is 2^k.

The next line contains 2^k space-separated integers $a_1, a_2, \ldots, a_{2^k}$ ($0 \le a_i < 2^k$) — the elements of the given array.

Output

If the given array can't be represented as element-wise XOR of two permutations of integers $0, 1, \ldots, 2^k - 1$, print "Fou".

Otherwise, print "Shi" in the first line.

The next two lines should contain the description of two suitable permutations. The first of these lines should contain 2^k space-separated distinct integers $p_1, p_2, \ldots, p_{2^k}$, and the second line should contain 2^k space-separated distinct integers $q_1, q_2, \ldots, q_{2^k}$.

All elements of p and q should be between 0 and $2^k - 1$, inclusive; $p_i \oplus q_i$ should be equal to a_i for all i such that $1 \le i \le 2^k$. If there are several possible solutions, you can print any.

Examples

input
```
2
0 1 2 3
```
output
```
Shi
2 0 1 3
2 1 3 0
```

input
```
2
0 0 0 0
```
output
```
Shi
0 1 2 3
0 1 2 3
```

input
```
2
0 1 2 2
```
output
```
Fou
```