Problem A. New Year Train

Input file:	a.in
Output file:	a.out
Time limit:	2 seconds
Memory limit:	128 megabytes

On the New Year Eve a government of one country decided to send a train with gifts to each town of the country. For each of N towns exactly one wagon with gifts was sent. The route was organized in such way that at each place the last wagon would be detached, then in the next place the other last one and so on. Just before the departure it turned out that the loading workers did not pay attention to numeration of the wagons and loaded the gifts into the wagons in random order. It was impossible to detach a wagon from the middle of the train and there was no time to rearrange gifts.

Luckily, there was a depot with parallel tracks. At the entrance of the depot each wagon could be directed to any of the tracks, and wagons could leave the depot from the other side in the right sequences: 1, 2, 3, 4, and so on.

For example, if at the entrance of the depot with three parallel tracks there were six wagons standing in order: 2, 5, 1, 4, 6, 3, then wagons 2, 5, 6 could be directed to the first track; wagons 1, 4 to the second track, wagon 3 to the third track could be directed respectively. In this case wagons could leave the depot in the right order.

Fortunately, there were enough tracks in the depot to rearrange the train.

Input

The first line of input file contains two integers N and M — the number of wagons in the train and the number of tracks in the depot respectively ($1 \le N \le 800\,000$, $1 \le M \le 100\,000$, $M \le N$). The second line contains N integers — sequence of wagons before the entrance to the depot.

It's guaranteed that solution always exists.

Output

On the first line output file must contain N integers — which track should be chosen for each wagon from input data. On the second line of output file write the number of tracks in order the wagons should leave the depot in the sequence 1, 2, 3, and so on.

Examples

a.in	a.out
6 3 2 5 1 4 6 3	1 1 2 2 1 3 2 1 3 2 1 1

Note

In 30% of testcases $N \le 100$.

In 60% of testcases $N \le 10\,000$.

Problem B. Beautiful row

Input file:	b.in
Output file:	b.out
Time limit:	3 seconds
Memory limit:	256 megabytes

Ali-Amir wrote N numbers in a row. A row is called beautiful if any two of the neighbour numbers in the row have got the same amount of ones in binary or ternary notations.

Ali-Amir wants to count the number of ways the all given numbers can be written in a beautiful row.

Input

The first line of input file contains integer N ($2 \leq N \leq 20$). The next line contains N non-negative integers not exceeding 10^9 each.

Output

Output the number of ways the all given numbers can be placed in a beautiful row.

Examples

b.in	b.out
3 5 1 6	2

Note

In the sample $5 = 12_3$ and $1 = 1_3$, $5 = 101_2$ and $6 = 110_2$, thus rows 1 5 6 and 6 5 1 are beautiful.

In 25% of testcases $N \leq 4$.

In 50% of testcases $N \leq 10$.

Problem C.

Input file:	`c.in`
Output file:	`c.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given numbers N, x and a sequence of N numbers. Find the largest possible interval of consequently following elements, such that "xor"of these elements is not less than x. I.e., more formally, find such i and k that

$a_i \oplus a_{i+1} \oplus \cdots \oplus a_{i+k-1} \geq x$, $1 \leq i \leq i+k-1 \leq N$,

and k is largest possible positive number.

It's guaranteed that for each test from the testset such an interval exists.

We remind you that $xor(\oplus)$ operation is applied to numbers in binary representation, so that for each pair of bits the following is true:

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

The result of this operation doesn't depend on the order of operands $a \oplus b = b \oplus a$. Moreover $(a \oplus (a \oplus b)) = b$.

In Pascal this operation is represented as xor. In C/C++/Java as $\wedge$.

Input

The first line of input contains N ($1 \leq N \leq 250\,000$) and x ($0 \leq x \leq 1\,000\,000\,000$). The second line of input contains N non-negative numbers not exceeding 10^9.

Output

The first line of output must contain two numbers: i and k. In case of many solutions output the one with the smallest i.

Examples

c.in	c.out
4 6 6 1 2 4	2 3

Note

In 40% testcases $N \leq 35\,000$.

In 80% testcases $N \leq 100\,000$.

Problem D. Biochips

Input file:	d.in
Output file:	d.out
Time limit:	2 seconds
Memory limit:	512 megabytes

Scientists discovered a biochip, which can divide itself into several new biochips. Herewith the parent-biochip disappears. Each biochip has its own size of memory, the size does not depend on the parent memory size. Then the biochip might be either used (stopping its division), or it will continue division in a similar manner.

Scientists prepared tree-like schemes of biochip-division process and they know the exact structure of the tree consisting of N elements, including the memory size of each of possible biochips descendants.

Make a program to choose from the tree exactly M biochips total memory size of which is the biggest possible. Pay attention to the fact that when we are choosing a biochip, we can't choose neither any of its ancestors nor descendants.

Input

The first line of input file contains two integers N and M — number of elements of the tree and number of biochips to be chosen ($1 \le N \le 200\,000$, $1 \le M \le 500$).

The following N lines contain two non-negative integers each: number of the parent in the tree and the size of the chip's memory x ($1 \le x \le 1000$). Each biochip has a unique number ranging from 1 to N inclusively. The line with i number includes information about a biochip with number $i - 1$. Just one biochip doesn't have a parent and it's parent is written as 0.

It's guaranteed that it's possible to choose M biochips.

Output

Output file consists of one integer — maximal possible amount of memory of M chosen biochips.

Examples

d.in	d.out
7 3 2 100 0 1000 2 150 3 100 3 5 5 100 2 50	300

Note

In 20% of testcases $N \le 20, M \le 10$.

In 60% of testcases $N \le 10\,000, M \le 100$.