## VK Cup 2012 Round 3 (Unofficial Div. 2 Edition)

# A. Problem About Equation

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

A group of $n$ merry programmers celebrate Robert Floyd's birthday. Polucarpus has got an honourable task of pouring Ber-Cola to everybody. Pouring the same amount of Ber-Cola to everybody is really important. In other words, the drink's volume in each of the $n$ mugs must be the same.

Polycarpus has already began the process and he partially emptied the Ber-Cola bottle. Now the first mug has $a_1$ milliliters of the drink, the second one has $a_2$ milliliters and so on. The bottle has $b$ milliliters left and Polycarpus plans to pour them into the mugs so that the main equation was fulfilled.

Write a program that would determine what volume of the drink Polycarpus needs to add into each mug to ensure that the following two conditions were fulfilled simultaneously:

- there were $b$ milliliters poured in total. That is, the bottle need to be emptied;
- after the process is over, the volumes of the drink in the mugs should be equal.

### Input

The first line contains a pair of integers $n$, $b$ ($2 \le n \le 100$, $1 \le b \le 100$), where $n$ is the total number of friends in the group and $b$ is the current volume of drink in the bottle. The second line contains a sequence of integers $a_1$, $a_2$, ..., $a_n$ ($0 \le a_i \le 100$), where $a_i$ is the current volume of drink in the $i$-th mug.

### Output

Print a single number "-1" (without the quotes), if there is no solution. Otherwise, print $n$ float numbers $c_1$, $c_2$, ..., $c_n$, where $c_i$ is the volume of the drink to add in the $i$-th mug. Print the numbers with no less than 6 digits after the decimal point, print each $c_i$ on a single line. Polycarpus proved that if a solution exists then it is unique.

Russian locale is installed by default on the testing computer. Make sure that your solution use the point to separate the integer part of a real number from the decimal, not a comma.

**Sample test(s)**

| input |
| --- |
| 5 50<br>1 2 3 4 5 |
| output |
| 12.000000<br>11.000000<br>10.000000<br>9.000000<br>8.000000 |

| input |
| --- |
| 2 2<br>1 100 |
| output |
| -1 |

# B. File List

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

Eudokimus, a system administrator is in trouble again. As a result of an error in some script, a list of names of very important files has been damaged. Since they were files in the BerFS file system, it is known that each file name has a form "`name.ext`", where:

- `name` is a string consisting of lowercase Latin letters, its length is from 1 to 8 characters;
- `ext` is a string consisting of lowercase Latin letters, its length is from 1 to 3 characters.

For example, "`read.me`", "`example.txt`" and "`b.cpp`" are valid file names and "`version.info`", "`ntldr`" and "`contestdata.zip`" are not.

Damage to the list meant that all the file names were recorded one after another, without any separators. So now Eudokimus has a single string.

Eudokimus needs to set everything right as soon as possible. He should divide the resulting string into parts so that each part would be a valid file name in BerFS. Since Eudokimus has already proved that he is not good at programming, help him. The resulting file list can contain the same file names.

## Input

The input data consists of a single string $s$, its length is from $1$ to $4 \cdot 10^5$ characters. The string can contain only lowercase Latin letters ('`a`' - '`z`') and periods ('`.`').

## Output

In the first line print "`YES`" (without the quotes), if it is possible to divide $s$ into parts as required. In this case, the following lines should contain the parts of the required partition, one per line in the order in which they appear in $s$. The required partition can contain the same file names. If there are multiple solutions, print any of them.

If the solution does not exist, then print in a single line "`NO`" (without the quotes).

**Sample test(s)**

| input |
| --- |
| `read.meexample.txtb.cpp` |
| output |
| `YES`<br>`read.m`<br>`eexample.t`<br>`xtb.cpp` |

| input |
| --- |
| `version.infontldrcontestdata.zip` |
| output |
| `NO` |

# C. Range Increments

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

Polycarpus is an amateur programmer. Now he is analyzing a friend's program. He has already found there the function `rangeIncrement(l, r)`, that adds `1` to each element of some array $a$ for all indexes in the segment $[l, r]$. In other words, this function does the following:

```
function rangeIncrement(l, r)
    for i := l .. r do
        a[i] = a[i] + 1
```

Polycarpus knows the state of the array $a$ after a series of function calls. He wants to determine the minimum number of function calls that lead to such state. In addition, he wants to find what function calls are needed in this case. It is guaranteed that the required number of calls does not exceed $10^5$.

Before calls of function `rangeIncrement(l, r)` all array elements equal zero.

## Input

The first input line contains a single integer $n$ ($1 \le n \le 10^5$) — the length of the array $a[1... n]$.

The second line contains its integer space-separated elements, $a[1], a[2], ..., a[n]$ ($0 \le a[i] \le 10^5$) after some series of function calls `rangeIncrement(l, r)`.

It is guaranteed that at least one element of the array is positive. It is guaranteed that the answer contains no more than $10^5$ calls of function `rangeIncrement(l, r)`.

## Output

Print on the first line $t$ — the minimum number of calls of function `rangeIncrement(l, r)`, that lead to the array from the input data. It is guaranteed that this number will turn out not more than $10^5$.

Then print $t$ lines — the descriptions of function calls, one per line. Each line should contain two integers $l_i, r_i$ ($1 \le l_i \le r_i \le n$) — the arguments of the $i$-th call `rangeIncrement(l, r)`. Calls can be applied in any order.

If there are multiple solutions, you are allowed to print any of them.

**Sample test(s)**

| input |
| --- |
| 6<br>1 2 1 1 4 1 |

| output |
| --- |
| 5<br>2 2<br>5 5<br>5 5<br>5 5<br>1 6 |

| input |
| --- |
| 5<br>1 0 1 0 1 |

| output |
| --- |
| 3<br>1 1<br>3 3<br>5 5 |

## Note

The first sample requires a call for the entire array, and four additional calls:

- one for the segment [2,2] (i.e. the second element of the array),
- three for the segment [5,5] (i.e. the fifth element of the array).

# D. Variable, or There and Back Again

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

Life is not easy for the perfectly common variable named Vasya. Wherever it goes, it is either assigned a value, or simply ignored, or is being used!

Vasya's life goes in states of a program. In each state, Vasya can either be used (for example, to calculate the value of another variable), or be assigned a value, or ignored. Between some states are directed (oriented) transitions.

A *path* is a sequence of states $v_1, v_2, ..., v_x$, where for any $1 \le i < x$ exists a transition from $v_i$ to $v_{i+1}$.

Vasya's value in state $v$ is interesting to the world, if exists path $p_1, p_2, ..., p_k$ such, that $p_i = v$ for some $i$ $(1 \le i \le k)$, in state $p_1$ Vasya gets assigned a value, in state $p_k$ Vasya is used and there is no state $p_i$ (except for $p_1$) where Vasya gets assigned a value.

Help Vasya, find the states in which Vasya's value is interesting to the world.

## Input

The first line contains two space-separated integers $n$ and $m$ $(1 \le n, m \le 10^5)$ — the numbers of states and transitions, correspondingly.

The second line contains space-separated $n$ integers $f_1, f_2, ..., f_n$ $(0 \le f_i \le 2)$, $f_i$ described actions performed upon Vasya in state $i$: $0$ represents ignoring, $1$ — assigning a value, $2$ — using.

Next $m$ lines contain space-separated pairs of integers $a_i, b_i$ $(1 \le a_i, b_i \le n, a_i \ne b_i)$, each pair represents the transition from the state number $a_i$ to the state number $b_i$. Between two states can be any number of transitions.

## Output

Print $n$ integers $r_1, r_2, ..., r_n$, separated by spaces or new lines. Number $r_i$ should equal $1$, if Vasya's value in state $i$ is interesting to the world and otherwise, it should equal $0$. The states are numbered from $1$ to $n$ in the order, in which they are described in the input.

**Sample test(s)**

input

```
4 3
1 0 0 2
1 2
2 3
3 4
```

output

```
1
1
1
1
```

input

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

output

```
1
0
1
```

input

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

output

```
0
0
0
```

## Note

In the first sample the program states can be used to make the only path in which the value of Vasya interests the world, 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4; it includes all the states, so in all of them Vasya's value is interesting to the world.

The second sample the only path in which Vasya's value is interesting to the world is , — 1 $\rightarrow$ 3; state 2 is not included there.

In the third sample we cannot make from the states any path in which the value of Vasya would be interesting to the world, so the value of Vasya is never interesting to the world.

# E. Ancient Berland Hieroglyphs

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarpus enjoys studying Berland hieroglyphs. Once Polycarp got hold of two ancient Berland pictures, on each of which was drawn a circle of hieroglyphs. We know that no hieroglyph occurs twice in either the first or the second circle (but in can occur once in each of them).

Polycarpus wants to save these pictures on his laptop, but the problem is, laptops do not allow to write hieroglyphs circles. So Polycarp had to break each circle and write down all of its hieroglyphs in a clockwise order in one line. A line obtained from the first circle will be called $a$, and the line obtained from the second one will be called $b$.

There are quite many ways to break hieroglyphic circles, so Polycarpus chooses the method, that makes the length of the largest substring of string $a$, which occurs as a subsequence in string $b$, maximum.

Help Polycarpus — find the maximum possible length of the desired substring (subsequence) if the first and the second circles are broken optimally.

The *length* of string $s$ is the number of characters in it. If we denote the length of string $s$ as $|s|$, we can write the string as $s = s_1 s_2 \ldots s_{|s|}$.

A *substring* of $s$ is a non-empty string $x = s[a \ldots b] = s_a s_{a+1} \ldots s_b$ $(1 \le a \le b \le |s|)$. For example, "`code`" and "`force`" are substrings of "`codeforces`", while "`coders`" is not.

A *subsequence* of $s$ is a non-empty string $y = s[p_1 p_2 \ldots p_{|y|}] = s_{p_1} s_{p_2} \ldots s_{p_{|y|}}$ $(1 \le p_1 < p_2 < \ldots < p_{|y|} \le |s|)$. For example, "`coders`" is a subsequence of "`codeforces`".

## Input

The first line contains two integers $l_a$ and $l_b$ $(1 \le l_a, l_b \le 1000000)$ — the number of hieroglyphs in the first and second circles, respectively.

Below, due to difficulties with encoding of Berland hieroglyphs, they are given as integers from $1$ to $10^6$.

The second line contains $l_a$ integers — the hieroglyphs in the first picture, in the clockwise order, starting with one of them.

The third line contains $l_b$ integers — the hieroglyphs in the second picture, in the clockwise order, starting with one of them.

It is guaranteed that the first circle doesn't contain a hieroglyph, which occurs twice. The second circle also has this property.

## Output

Print a single number — the maximum length of the common substring and subsequence. If at any way of breaking the circles it does not exist, print `0`.

**Sample test(s)**

| input |
| --- |
| 5 4<br>1 2 3 4 5<br>1 3 5 6 |
| output |
| 2 |

| input |
| --- |
| 4 6<br>1 3 5 2<br>1 2 3 4 5 6 |
| output |
| 3 |

| input |
| --- |
| 3 3<br>1 2 3<br>3 2 1 |
| output |
| 2 |

## Note

In the first test Polycarpus picks a string that consists of hieroglyphs 5 and 1, and in the second sample — from hieroglyphs 1, 3 and 5.