Codeforces Round #547 (Div. 3)

A. Game 23

1 second, 256 megabytes

Polycarp plays "Game 23". Initially he has a number n and his goal is to transform it to m. In one move, he can multiply n by 2 or multiply n by 3. He can perform any number of moves.

Print the number of moves needed to transform n to m. Print -1 if it is impossible to do so.

It is easy to prove that any way to transform n to m contains the same number of moves (i.e. number of moves doesn't depend on the way of transformation).

Input

The only line of the input contains two integers n and m ($1 \le n \le m \le 5 \cdot 10^8$).

Output

Print the number of moves to transform n to m, or -1 if there is no solution.

input
120 51840
output
7

input
42 42
output
0

input
48 72
output
-1

In the first example, the possible sequence of moves is: $120 \to 240 \to 720 \to 1440 \to 4320 \to 12960 \to 25920 \to 51840$. The are 7 steps in total.

In the second example, no moves are needed. Thus, the answer is 0.

In the third example, it is impossible to transform 48 to 72.

B. Maximal Continuous Rest

2 seconds, 256 megabytes

Each day in Berland consists of n hours. Polycarp likes time management. That's why he has a fixed schedule for each day — it is a sequence $a_1, a_2, \ldots, a_n$ (each a_i is either 0 or 1), where $a_i = 0$ if Polycarp works during the i-th hour of the day and $a_i = 1$ if Polycarp rests during the i-th hour of the day.

Days go one after another endlessly and Polycarp uses the same schedule for each day.

What is the maximal number of continuous hours during which Polycarp rests? It is guaranteed that there is at least one working hour in a day.

Input

The first line contains n ($1 \le n \le 2 \cdot 10^5$) — number of hours per day.

The second line contains n integer numbers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$), where $a_i = 0$ if the i-th hour in a day is working and $a_i = 1$ if the i-th hour is resting. It is guaranteed that $a_i = 0$ for at least one i.

Output

Print the maximal number of continuous hours during which Polycarp rests. Remember that you should consider that days go one after another endlessly and Polycarp uses the same schedule for each day.

input
5 1 0 1 0 1
output
2

input
6 0 1 0 1 1 0
output
2

input
7 1 0 1 1 1 0 1
output
3

input
3 0 0 0
output
0

In the first example, the maximal rest starts in last hour and goes to the first hour of the next day.

In the second example, Polycarp has maximal rest from the 4-th to the 5-th hour.

In the third example, Polycarp has maximal rest from the 3-rd to the 5-th hour.

In the fourth example, Polycarp has no rest at all.

C. Polycarp Restores Permutation

2 seconds, 256 megabytes

An array of integers $p_1, p_2, \ldots, p_n$ is called a *permutation* if it contains each number from 1 to n exactly once. For example, the following arrays are permutations: $[3, 1, 2]$, $[1]$, $[1, 2, 3, 4, 5]$ and $[4, 3, 1, 2]$. The following arrays are *not* permutations: $[2]$, $[1, 1]$, $[2, 3, 4]$.

Polycarp invented a really cool permutation $p_1, p_2, \ldots, p_n$ of length n. It is very disappointing, but he forgot this permutation. He only remembers the array $q_1, q_2, \ldots, q_{n-1}$ of length $n-1$, where $q_i = p_{i+1} - p_i$.

Given n and $q = q_1, q_2, \ldots, q_{n-1}$, help Polycarp restore the invented permutation.

Input
The first line contains the integer n ($2 \le n \le 2 \cdot 10^5$) — the length of the permutation to restore. The second line contains $n-1$ integers $q_1, q_2, \ldots, q_{n-1}$ ($-n < q_i < n$).

Output
Print the integer `-1` if there is no such permutation of length n which corresponds to the given array q. Otherwise, if it exists, print $p_1, p_2, \ldots, p_n$. Print any such permutation if there are many of them.

input
3 -2 1
output
3 1 2

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

input
4 -1 2 2
output
-1

D. Colored Boots

2 seconds, 256 megabytes

There are n left boots and n right boots. Each boot has a color which is denoted as a lowercase Latin letter or a question mark ('`?`'). Thus, you are given two strings l and r, both of length n. The character l_i stands for the color of the i-th left boot and the character r_i stands for the color of the i-th right boot.

A lowercase Latin letter denotes a specific color, but the question mark ('`?`') denotes an indefinite color. Two specific colors are *compatible* if they are exactly the same. An indefinite color is *compatible* with any (specific or indefinite) color.

For example, the following pairs of colors are compatible: ('`f`', '`f`'), ('`?`', '`z`'), ('`a`', '`?`') and ('`?`', '`?`'). The following pairs of colors are *not* compatible: ('`f`', '`g`') and ('`a`', '`z`').

Compute the maximum number of pairs of boots such that there is one left and one right boot in a pair and their colors are compatible.

Print the maximum number of such pairs and the pairs themselves. A boot can be part of at most one pair.

Input
The first line contains n ($1 \le n \le 150000$), denoting the number of boots for each leg (i.e. the number of left boots and the number of right boots).

The second line contains the string l of length n. It contains only lowercase Latin letters or question marks. The i-th character stands for the color of the i-th left boot.

The third line contains the string r of length n. It contains only lowercase Latin letters or question marks. The i-th character stands for the color of the i-th right boot.

Output
Print k — the maximum number of compatible left-right pairs of boots, i.e. pairs consisting of one left and one right boot which have compatible colors.

The following k lines should contain pairs a_j, b_j ($1 \le a_j, b_j \le n$). The j-th of these lines should contain the index a_j of the left boot in the j-th pair and index b_j of the right boot in the j-th pair. All the numbers a_j should be distinct (unique), all the numbers b_j should be distinct (unique).

If there are many optimal answers, print any of them.

input
10 codeforces dodivthree
output
5 7 8 4 9 2 2 9 10 3 1

input
7 abaca?b zabbbcc
output
5 6 5 2 3 4 6 7 4 1 2

input
9 bambarbia hellocode
output
0

input
10 code?????? ??????test

output

```
10
6 2
1 6
7 3
3 5
4 8
9 7
5 1
2 4
10 9
8 10
```

E. Superhero Battle

2 seconds, 256 megabytes

A superhero fights with a monster. The battle consists of rounds, each of which lasts exactly n minutes. After a round ends, the next round starts immediately. This is repeated over and over again.

Each round has the same scenario. It is described by a sequence of n numbers: $d_1, d_2, \ldots, d_n$ ($-10^6 \le d_i \le 10^6$). The i-th element means that monster's hp (hit points) changes by the value d_i during the i-th minute of each round. Formally, if before the i-th minute of a round the monster's hp is h, then after the i-th minute it changes to $h := h + d_i$.

The monster's initial hp is H. It means that before the battle the monster has H hit points. Print the first minute after which the monster dies. The monster dies if its hp is less than or equal to 0. Print `-1` if the battle continues infinitely.

Input

The first line contains two integers H and n ($1 \le H \le 10^{12}$, $1 \le n \le 2 \cdot 10^5$). The second line contains the sequence of integers $d_1, d_2, \ldots, d_n$ ($-10^6 \le d_i \le 10^6$), where d_i is the value to change monster's hp in the i-th minute of a round.

Output

Print `-1` if the superhero can't kill the monster and the battle will last infinitely. Otherwise, print the positive integer k such that k is the first minute after which the monster is dead.

input

```
1000 6
-100 -200 -300 125 77 -4
```

output

```
9
```

input

```
1000000000000 5
-1 0 0 0 0
```

output

```
4999999999996
```

input

```
10 4
-3 -6 5 4
```

output

```
-1
```

F1. Same Sum Blocks (Easy)

2 seconds, 256 megabytes

This problem is given in two editions, which differ exclusively in the constraints on the number n.

You are given an array of integers $a[1], a[2], \ldots, a[n]$. A *block* is a sequence of contiguous (consecutive) elements $a[l], a[l+1], \ldots, a[r]$ ($1 \le l \le r \le n$). Thus, a block is defined by a pair of indices (l, r).

Find a set of blocks $(l_1, r_1), (l_2, r_2), \ldots, (l_k, r_k)$ such that:

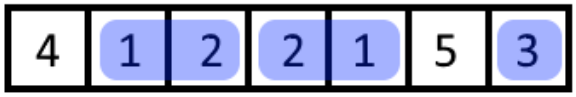
- They do not intersect (i.e. they are disjoint). Formally, for each pair of blocks (l_i, r_i) and (l_j, r_j) where $i \ne j$ either $r_i < l_j$ or $r_j < l_i$.
- For each block the sum of its elements is the same. Formally,

$$a[l_1] + a[l_1 + 1] + \cdots + a[r_1] = a[l_2] + a[l_2 + 1] + \cdots + a[r_2]$$

$$\cdots =$$

$$a[l_k] + a[l_k + 1] + \cdots + a[r_k].$$

- The number of the blocks in the set is maximum. Formally, there does not exist a set of blocks $(l'_1, r'_1), (l'_2, r'_2), \ldots, (l'_{k'}, r'_{k'})$ satisfying the above two requirements with $k' > k$.



The picture corresponds to the first example. Blue boxes illustrate blocks.

Write a program to find such a set of blocks.

Input

The first line contains integer n ($1 \le n \le 50$) — the length of the given array. The second line contains the sequence of elements $a[1], a[2], \ldots, a[n]$ ($-10^5 \le a_i \le 10^5$).

Output

In the first line print the integer k ($1 \le k \le n$). The following k lines should contain blocks, one per line. In each line print a pair of indices l_i, r_i ($1 \le l_i \le r_i \le n$) — the bounds of the i-th block. You can print blocks in any order. If there are multiple answers, print any of them.

input

```
7
4 1 2 2 1 5 3
```

output

```
3
7 7
2 3
4 5
```

input

```
11
-5 -4 -3 -2 -1 0 1 2 3 4 5
```

output

```
2
3 4
1 1
```

input

```
4
1 1 1 1
```

output

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

F2. Same Sum Blocks (Hard)

3 seconds, 256 megabytes

This problem is given in two editions, which differ exclusively in the constraints on the number n.

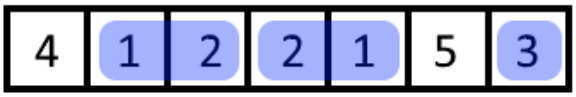
You are given an array of integers $a[1], a[2], \ldots, a[n]$. A *block* is a sequence of contiguous (consecutive) elements $a[l], a[l+1], \ldots, a[r]$ ($1 \le l \le r \le n$). Thus, a block is defined by a pair of indices (l, r).

Find a set of blocks $(l_1, r_1), (l_2, r_2), \ldots, (l_k, r_k)$ such that:

- They do not intersect (i.e. they are disjoint). Formally, for each pair of blocks (l_i, r_i) and (l_j, r_j) where $i \neq j$ either $r_i < l_j$ or $r_j < l_i$.
- For each block the sum of its elements is the same. Formally,

$$a[l_1] + a[l_1 + 1] + \cdots + a[r_1] = a[l_2] + a[l_2 + 1] + \cdots + a[r_2] = \cdots = a[l_k] + a[l_k + 1] + \cdots + a[r_k].$$

- The number of the blocks in the set is maximum. Formally, there does not exist a set of blocks $(l'_1, r'_1), (l'_2, r'_2), \ldots, (l'_{k'}, r'_{k'})$ satisfying the above two requirements with $k' > k$.



The picture corresponds to the first example. Blue boxes illustrate blocks.

Write a program to find such a set of blocks.

Input

The first line contains integer n ($1 \le n \le 1500$) — the length of the given array. The second line contains the sequence of elements $a[1], a[2], \ldots, a[n]$ ($-10^5 \le a_i \le 10^5$).

Output

In the first line print the integer k ($1 \le k \le n$). The following k lines should contain blocks, one per line. In each line print a pair of indices l_i, r_i ($1 \le l_i \le r_i \le n$) — the bounds of the i-th block. You can print blocks in any order. If there are multiple answers, print any of them.

input

```
7
4 1 2 2 1 5 3
```

output

```
3
7 7
2 3
4 5
```

input

```
11
-5 -4 -3 -2 -1 0 1 2 3 4 5
```

output

```
2
3 4
1 1
```

input

```
4
1 1 1 1
```

output

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

G. Privatization of Roads in Treeland

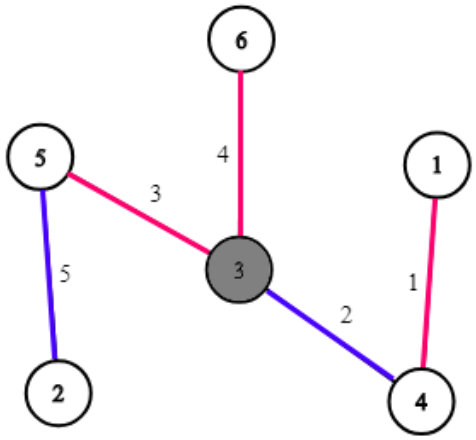
2 seconds, 256 megabytes

Treeland consists of n cities and $n - 1$ roads. Each road is bidirectional and connects two distinct cities. From any city you can get to any other city by roads. Yes, you are right — the country's topology is an undirected tree.

There are some private road companies in Treeland. The government decided to sell roads to the companies. Each road will belong to one company and a company can own multiple roads.

The government is afraid to look unfair. They think that people in a city can consider them unfair if there is one company which owns two or more roads entering the city. The government wants to make such privatization that the number of such cities doesn't exceed k and the number of companies taking part in the privatization is minimal.

Choose the number of companies r such that it is possible to assign each road to one company in such a way that the number of cities that have two or more roads of one company is at most k. In other words, if for a city all the roads belong to the different companies then the city is *good*. Your task is to find the minimal r that there is such assignment to companies from 1 to r that the number of cities which are not *good* doesn't exceed k.



The picture illustrates the first example ($n = 6, k = 2$). The answer contains $r = 2$ companies. Numbers on the edges denote edge indices. Edge colors mean companies: *red* corresponds to the first company, *blue* corresponds to the second company. The gray vertex (number 3) is *not good*. The number of such vertices (just one) doesn't exceed $k = 2$. It is impossible to have at most $k = 2$ not good cities in case of one company.

Input

The first line contains two integers n and k ($2 \le n \le 200000, 0 \le k \le n-1$) — the number of cities and the maximal number of cities which can have two or more roads belonging to one company.

The following $n-1$ lines contain roads, one road per line. Each line contains a pair of integers x_i, y_i ($1 \le x_i, y_i \le n$), where x_i, y_i are cities connected with the i-th road.

Output

In the first line print the required r ($1 \le r \le n-1$). In the second line print $n-1$ numbers $c_1, c_2, \ldots, c_{n-1}$ ($1 \le c_i \le r$), where c_i is the company to own the i-th road. If there are multiple answers, print any of them.

input

```
6 2
1 4
4 3
3 5
3 6
5 2
```

output

```
2
1 2 1 1 2
```

input

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

output

```
1
1 1 1
```

input

```
10 2
10 3
1 2
1 3
1 4
2 5
2 6
2 7
3 8
3 9
```

output

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