A. Petya and Java

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

Little Petya has recently started attending a programming club. Naturally he is facing the problem of choosing a programming language. After long considerations he realized that Java is the best choice. The main argument in favor of choosing Java was that it has a very large integer data type, called BigInteger.

But having attended several classes of the club, Petya realized that not all tasks require using the BigInteger type. It turned out that in some tasks it is much easier to use small data types. That's why a question arises: "Which integer type to use if one wants to store a positive integer n?"

Petya knows only 5 integer types:

1) **byte** occupies 1 byte and allows you to store numbers from -128 to 127

2) **short** occupies 2 bytes and allows you to store numbers from -32768 to 32767

3) **int** occupies 4 bytes and allows you to store numbers from -2147483648 to 2147483647

4) **long** occupies 8 bytes and allows you to store numbers from -9223372036854775808 to 9223372036854775807

5) **BigInteger** can store any integer number, but at that it is not a primitive type, and operations with it are much slower.

For all the types given above the boundary values are included in the value range.

From this list, Petya wants to choose the smallest type that can store a positive integer n. Since BigInteger works much slower, Peter regards it last. Help him.

Input

The first line contains a positive number n. It consists of no more than 100 digits and doesn't contain any leading zeros. The number n can't be represented as an empty string.

Please, do not use `%lld` specificator to read or write 64-bit integers in C++. It is preffered to use `cout` (also you may use `%I64d`).

Output

Print the first type from the list "`byte`, `short`, `int`, `long`, `BigInteger`", that can store the natural number n, in accordance with the data given above.

Examples

input
```
127
```
output
```
byte
```

input
```
130
```
output
```
short
```

input
```
123456789101112131415161718192021222324
```
output
```
BigInteger
```

B. Petya and Countryside

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

Little Petya often travels to his grandmother in the countryside. The grandmother has a large garden, which can be represented as a rectangle $1 \times n$ in size, when viewed from above. This rectangle is divided into n equal square sections. The garden is very unusual as each of the square sections possesses its own fixed height and due to the newest irrigation system we can create artificial rain above each section.

Creating artificial rain is an expensive operation. That's why we limit ourselves to creating the artificial rain only above one section. At that, the water from each watered section will flow into its neighbouring sections if their height does not exceed the height of the section. That is, for example, the garden can be represented by a 1×5 rectangle, where the section heights are equal to 4, 2, 3, 3, 2. Then if we create an artificial rain over any of the sections with the height of 3, the water will flow over all the sections, except the ones with the height of 4. See the illustration of this example at the picture:

As Petya is keen on programming, he decided to find such a section that if we create artificial rain above it, the number of watered sections will be maximal. Help him.

Input

The first line contains a positive integer n ($1 \le n \le 1000$). The second line contains n positive integers which are the height of the sections. All the numbers are no less than 1 and not more than 1000.

Output

Print a single number, the maximal number of watered sections if we create artificial rain above exactly one section.

Examples

input

```
1
2
```

output

```
1
```

input

```
5
1 2 1 2 1
```

output

```
3
```

input

```
8
1 2 1 1 1 3 3 4
```

output

```
6
```

C. Petya and File System

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

Recently, on a programming lesson little Petya showed how quickly he can create files and folders on the computer. But he got soon fed up with this activity, and he decided to do a much more useful thing. He decided to calculate what folder contains most subfolders (including nested folders, nested folders of nested folders, and so on) and what folder contains most files (including the files in the subfolders).

More formally, the subfolders of the folder are all its directly nested folders and the subfolders of these nested folders. The given folder is not considered the subfolder of itself. A file is regarded as lying in a folder, if and only if it either lies directly in this folder, or lies in some subfolder of the folder.

For a better understanding of how to count subfolders and files for calculating the answer, see notes and answers to the samples.

You are given a few files that Petya has managed to create. The path to each file looks as follows:

$diskName$:\\$folder_1$\\$folder_2$\\...\\ $folder_n$\\$fileName$

- $diskName$ is single capital letter from the set `{C,D,E,F,G}`.
- $folder_1$, ..., $folder_n$ are folder names. Each folder name is nonempty sequence of lowercase Latin letters and digits from 0 to 9. ($n \geq 1$)
- $fileName$ is a file name in the form of $name.extension$, where the $name$ and the $extension$ are nonempty sequences of lowercase Latin letters and digits from 0 to 9.

It is also known that there is no file whose path looks like $diskName$:\\$fileName$. That is, each file is stored in some folder, but there are no files directly in the root. Also let us assume that the disk root is not a folder.

Help Petya to find the largest number of subfolders, which can be in some folder, and the largest number of files that can be in some folder, counting all its subfolders.

Input

Each line of input data contains the description of one file path. The length of each line does not exceed 100, and overall there are no more than 100 lines. It is guaranteed, that all the paths are correct and meet the above rules. It is also guaranteed, that there are no two completely equal lines. That is, each file is described exactly once.

There is at least one line in the input data.

Output

Print two space-separated numbers. The first one is the maximal number of possible subfolders in a folder (including nested folders, nested folders of nested folders, and so on). The second one is the maximal number of files in a folder (including nested files in subfolders). Note that the disks are not regarded as folders.

Examples

input

```
C:\folder1\file1.txt
```

output

```
0 1
```

input

```
C:\folder1\folder2\folder3\file1.txt
C:\folder1\folder2\folder4\file1.txt
D:\folder1\file1.txt
```

output

```
3 2
```

input

```
C:\file\file\file\file\file.txt
C:\file\file\file\file2\file.txt
```

output

```
4 2
```

Note

In the first sample we have one folder on the "C" disk. It has no subfolders, which is why the first number in the answer is 0. But this folder contains one file, so the second number of the answer is 1.

In the second sample we have several different folders. Consider the "`folder1`" folder on the "`C`" disk. This folder directly contains one folder, "`folder2`". The "`folder2`" folder contains two more folders — "`folder3`" and "`folder4`". Thus, the "`folder1`" folder on the "`C`" drive has exactly 3 subfolders. Also this folder contains two files, even though they do not lie directly in the folder, but they are located in subfolders of "`folder1`".

In the third example we see that the names of some folders and some subfolders are identical. Consider the "`file`" folder, which lies directly on the "`C`" disk. That folder contains another "`file`" folder, which in turn contains another "`file`" folder, which contains two more folders, "`file`" and "`file2`". Thus, the "`file`" folder, which lies directly on the "`C`" disk, contains 4 subfolders.

D. Petya and His Friends

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

Little Petya has a birthday soon. Due this wonderful event, Petya's friends decided to give him sweets. The total number of Petya's friends equals to n.

Let us remind you the definition of the greatest common divisor: $GCD(a_1, ..., a_k) = d$, where d represents such a maximal positive number that each a_i ($1 \le i \le k$) is evenly divisible by d. At that, we assume that all a_i's are greater than zero.

Knowing that Petya is keen on programming, his friends has agreed beforehand that the 1-st friend gives a_1 sweets, the 2-nd one gives a_2 sweets, ..., the n-th one gives a_n sweets. At the same time, for any i and j ($1 \le i, j \le n$) they want the $GCD(a_i, a_j)$ not to be equal to 1. However, they also want the following condition to be satisfied: $GCD(a_1, a_2, ..., a_n) = 1$. One more: all the a_i should be distinct.

Help the friends to choose the suitable numbers $a_1, ..., a_n$.

Input

The first line contains an integer n ($2 \le n \le 50$).

Output

If there is no answer, print "`-1`" without quotes. Otherwise print a set of n distinct positive numbers $a_1, a_2, ..., a_n$. Each line must contain one number. Each number must consist of not more than 100 digits, and must not contain any leading zeros. If there are several solutions to that problem, print any of them.

Do not forget, please, that all of the following conditions must be true:

- For every i and j ($1 \le i, j \le n$): $GCD(a_i, a_j) \ne 1$
- $GCD(a_1, a_2, ..., a_n) = 1$
- For every i and j ($1 \le i, j \le n, i \ne j$): $a_i \ne a_j$

Please, do not use `%lld` specificator to read or write 64-bit integers in C++. It is preffered to use `cout` (also you may use `%I64d`).

Examples

input

```
3
```

output

```
99
55
11115
```

input

```
4
```

output

```
385
360
792
8360
```

E. Petya and Post

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

Little Vasya's uncle is a postman. The post offices are located on one circular road. Besides, each post office has its own gas station located next to it. Petya's uncle works as follows: in the morning he should leave the house and go to some post office. In the office he receives a portion of letters and a car. Then he must drive in the given car exactly one round along the circular road and return to the starting post office (the uncle can drive along the circle in any direction, counterclockwise or clockwise). Besides, since the car belongs to the city post, it should also be fuelled with gasoline only at the Post Office stations.

The total number of stations equals to n. One can fuel the car at the i-th station with no more than a_i liters of gasoline. Besides, one can fuel the car no more than once at each station. Also, the distance between the 1-st and the 2-nd station is b_1 kilometers, the distance between the 2-nd and the 3-rd one is b_2 kilometers, ..., between the $(n - 1)$-th and the n-th ones the distance is $b_{n - 1}$ kilometers and between the n-th and the 1-st one the distance is b_n kilometers. Petya's uncle's high-tech car uses only one liter of gasoline per kilometer. It is known that the stations are located so that the sum of all a_i is equal to the sum of all b_i. The i-th gas station and i-th post office are very close, so the distance between them is 0 kilometers.

Thus, it becomes clear that if we start from some post offices, then it is not always possible to drive one round along a circular road. The uncle faces the following problem: to what stations can he go in the morning to be able to ride exactly one circle along the circular road and visit all the post offices that are on it?

Petya, who used to attend programming classes, has volunteered to help his uncle, but his knowledge turned out to be not enough, so he asks you to help him write the program that will solve the posed problem.

Input

The first line contains integer n ($1 \le n \le 10^5$). The second line contains n integers a_i — amount of gasoline on the i-th station. The third line contains n integers $b_1, b_2, ..., b_n$. They are the distances between the 1-st and the 2-nd gas stations, between the 2-nd and the 3-rd ones, ..., between the n-th and the 1-st ones, respectively. The sum of all b_i equals to the sum of all a_i and is no more than 10^9. Each of the numbers a_i, b_i is no less than 1 and no more than 10^9.

Output

Print on the first line the number k — the number of possible post offices, from which the car can drive one circle along a circular road. Print on the second line k numbers in the ascending order — the numbers of offices, from which the car can start.

Examples

input

```
4
1 7 2 3
8 1 1 3
```

output

```
2
2 4
```

input

```
8
1 2 1 2 1 2 1 2
2 1 2 1 2 1 2 1
```

output

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