A. Business trip

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

What joy! Petya's parents went on a business trip for the whole year and the playful kid is left all by himself. Petya got absolutely happy. He jumped on the bed and threw pillows all day long, until...

Today Petya opened the cupboard and found a scary note there. His parents had left him with duties: he should water their favourite flower all year, each day, in the morning, in the afternoon and in the evening. "Wait a second!" — thought Petya. He know for a fact that if he fulfills the parents' task in the i-th ($1 \le i \le 12$) month of the year, then the flower will grow by a_i centimeters, and if he doesn't water the flower in the i-th month, then the flower won't grow this month. Petya also knows that try as he might, his parents won't believe that he has been watering the flower if it grows strictly less than by k centimeters.

Help Petya choose the minimum number of months when he will water the flower, given that the flower should grow no less than by k centimeters.

Input

The first line contains exactly one integer k ($0 \le k \le 100$). The next line contains twelve space-separated integers: the i-th ($1 \le i \le 12$) number in the line represents a_i ($0 \le a_i \le 100$).

Output

Print the only integer — the minimum number of months when Petya has to water the flower so that the flower grows no less than by k centimeters. If the flower can't grow by k centimeters in a year, print -1.

Examples

input

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

output

```
2
```

input

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

output

```
0
```

input

```
11
1 1 4 1 1 5 1 1 4 1 1 1
```

output

```
3
```

Note

Let's consider the first sample test. There it is enough to water the flower during the seventh and the ninth month. Then the flower grows by exactly five centimeters.

In the second sample Petya's parents will believe him even if the flower doesn't grow at all ($k = 0$). So, it is possible for Petya not to water the flower at all.

B. Martian Clock

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

Having stayed home alone, Petya decided to watch forbidden films on the Net in secret. "What ungentlemanly behavior!" — you can say that, of course, but don't be too harsh on the kid. In his country films about the Martians and other extraterrestrial civilizations are forbidden. It was very unfair to Petya as he adored adventure stories that featured lasers and robots.

Today Petya is watching a shocking blockbuster about the Martians called "R2:D2". What can "R2:D2" possibly mean? It might be the Martian time represented in the Martian numeral system. Petya knows that time on Mars is counted just like on the Earth (that is, there are 24 hours and each hour has 60 minutes). The time is written as "a:b", where the string a stands for the number of hours (from 0 to 23 inclusive), and string b stands for the number of minutes (from 0 to 59 inclusive). The only thing Petya doesn't know is in what numeral system the Martian time is written.

Your task is to print the radixes of all numeral system which can contain the time "a:b".

Input

The first line contains a single string as "a:b" (without the quotes). There a is a non-empty string, consisting of numbers and uppercase Latin letters. String a shows the number of hours. String b is a non-empty string that consists of numbers and uppercase Latin letters. String b shows the number of minutes. The lengths of strings a and b are from 1 to 5 characters, inclusive. Please note that strings a and b can have leading zeroes that do not influence the result in any way (for example, string "`008:1`" in decimal notation denotes correctly written time).

We consider characters `0`, `1`, ..., `9` as denoting the corresponding digits of the number's representation in some numeral system, and characters `A`, `B`, ..., `Z` correspond to numbers `10`, `11`, ..., `35`.

Output

Print the radixes of the numeral systems that can represent the time "a:b" in the increasing order. Separate the numbers with spaces or line breaks. If there is no numeral system that can represent time "a:b", print the single integer `0`. If there are infinitely many numeral systems that can represent the time "a:b", print the single integer `-1`.

Note that on Mars any positional numeral systems with positive radix strictly larger than one are possible.

Examples

input
```
11:20
```
output
```
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

input
```
2A:13
```
output
```
0
```

input
```
000B:00001
```
output
```
-1
```

Note

Let's consider the first sample. String "`11:20`" can be perceived, for example, as time `4:6`, represented in the ternary numeral system or as time `17:32` in hexadecimal system.

Let's consider the second sample test. String "`2A:13`" can't be perceived as correct time in any notation. For example, let's take the base-11 numeral notation. There the given string represents time `32:14` that isn't a correct time.

Let's consider the third sample. String "`000B:00001`" can be perceived as a correct time in the infinite number of numeral systems. If you need an example, you can take any numeral system with radix no less than 12.

C. Division into Teams

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

Petya loves football very much, especially when his parents aren't home. Each morning he comes to the yard, gathers his friends and they play all day. From time to time they have a break to have some food or do some chores (for example, water the flowers).

The key in football is to divide into teams fairly before the game begins. There are n boys playing football in the yard (including Petya), each boy's football playing skill is expressed with a non-negative characteristic a_i (the larger it is, the better the boy plays).

Let's denote the number of players in the first team as x, the number of players in the second team as y, the individual numbers of boys who play for the first team as p_i and the individual numbers of boys who play for the second team as q_i. Division n boys into two teams is considered fair if three conditions are fulfilled:

- Each boy plays for exactly one team ($x + y = n$).
- The sizes of teams differ in no more than one ($|x - y| \le 1$).
- The total football playing skills for two teams differ in no more than by the value of skill the best player in the yard has. More formally:

$$|\sum_{i=1}^{x} a_{p_i} - \sum_{i=1}^{y} a_{q_i}| \le \max_{i=1}^{n} a_i$$

Your task is to help guys divide into two teams fairly. It is guaranteed that a fair division into two teams always exists.

Input

The first line contains the only integer n ($2 \le n \le 10^5$) which represents the number of guys in the yard. The next line contains n positive space-separated integers, a_i ($1 \le a_i \le 10^4$), the i-th number represents the i-th boy's playing skills.

Output

On the first line print an integer x — the number of boys playing for the first team. On the second line print x integers — the individual numbers of boys playing for the first team. On the third line print an integer y — the number of boys playing for the second team, on the fourth line print y integers — the individual numbers of boys playing for the second team. Don't forget that you should fulfil all three conditions: $x + y = n$, $|x - y| \le 1$, and the condition that limits the total skills.

If there are multiple ways to solve the problem, print any of them.

The boys are numbered starting from one in the order in which their skills are given in the input data. You are allowed to print individual numbers of boys who belong to the same team in any order.

Examples

input

```
3
1 2 1
```

output

```
2
1 2
1
3
```

input

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

output

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

Note

Let's consider the first sample test. There we send the first and the second boy to the first team and the third boy to the second team. Let's check all three conditions of a fair division. The first limitation is fulfilled (all boys play), the second limitation on the sizes of groups ($|2 - 1| = 1 \le 1$) is fulfilled, the third limitation on the difference in skills ($(2 + 1) - (1) = 2 \le 2$) is fulfilled.

D. Coloring Brackets

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

Once Petya read a problem about a bracket sequence. He gave it much thought but didn't find a solution. Today you will face it.

You are given string s. It represents a correct bracket sequence. A correct bracket sequence is the sequence of opening ("`(`") and closing ("`)`") brackets, such that it is possible to obtain a correct mathematical expression from it, inserting numbers and operators between the brackets. For example, such sequences as "`(())()`" and "`()`" are correct bracket sequences and such sequences as "`)()`" and "`(()`" are not.

In a correct bracket sequence each bracket corresponds to the matching bracket (an opening bracket corresponds to the matching closing bracket and vice versa). For example, in a bracket sequence shown of the figure below, the third bracket corresponds to the matching sixth one and the fifth bracket corresponds to the fourth one.

You are allowed to color some brackets in the bracket sequence so as all three conditions are fulfilled:

- Each bracket is either not colored any color, or is colored red, or is colored blue.
- For any pair of matching brackets exactly one of them is colored. In other words, for any bracket the following is true: either it or the matching bracket that corresponds to it is colored.
- No two neighboring colored brackets have the same color.

Find the number of different ways to color the bracket sequence. The ways should meet the above-given conditions. Two ways of coloring are considered different if they differ in the color of at least one bracket. As the result can be quite large, print it modulo 1000000007 ($10^9 + 7$).

Input

The first line contains the single string s ($2 \le |s| \le 700$) which represents a correct bracket sequence.

Output

Print the only number — the number of ways to color the bracket sequence that meet the above given conditions modulo 1000000007 ($10^9 + 7$).

Examples

input
```
(())
```
output
```
12
```

input
```
(()())
```
output
```
40
```

input
```
()
```
output
```
4
```

Note

Let's consider the first sample test. The bracket sequence from the sample can be colored, for example, as is shown on two figures below.

The two ways of coloring shown below are incorrect.

E. Martian Strings

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

During the study of the Martians Petya clearly understood that the Martians are absolutely lazy. They like to sleep and don't like to wake up.

Imagine a Martian who has exactly n eyes located in a row and numbered from the left to the right from 1 to n. When a Martian sleeps, he puts a patch on each eye (so that the Martian morning doesn't wake him up). The inner side of each patch has an uppercase Latin letter. So, when a Martian wakes up and opens all his eyes he sees a string s consisting of uppercase Latin letters. The string's length is n.

"Ding dong!" — the alarm goes off. A Martian has already woken up but he hasn't opened any of his eyes. He feels that today is going to be a hard day, so he wants to open his eyes and see something good. The Martian considers only m Martian words beautiful. Besides, it is hard for him to open all eyes at once so early in the morning. So he opens two non-overlapping segments of consecutive eyes. **More formally, the Martian chooses four numbers a, b, c, d, ($1 \le a \le b < c \le d \le n$) and opens all eyes with numbers i such that $a \le i \le b$ or $c \le i \le d$.** After the Martian opens the eyes he needs, he reads all the visible characters from the left to the right and thus, he sees some word.

Let's consider all different words the Martian can see in the morning. Your task is to find out how many beautiful words are among them.

Input

The first line contains a non-empty string s consisting of uppercase Latin letters. The strings' length is n ($2 \le n \le 10^5$). The second line contains an integer m ($1 \le m \le 100$) — the number of beautiful words. Next m lines contain the beautiful words p_i, consisting of uppercase Latin letters. Their length is from 1 to 1000. All beautiful strings are pairwise different.

Output

Print the single integer — the number of different beautiful strings the Martian can see this morning.

Examples

input

```
ABCBABA
2
BAAB
ABBA
```

output

```
1
```

Note

Let's consider the sample test. There the Martian can get only the second beautiful string if he opens segments of eyes $a = 1, b = 2$ and $c = 4, d = 5$ or of he opens segments of eyes $a = 1, b = 2$ and $c = 6, d = 7$.