Testing Round #9

A. Second-Price Auction

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem we consider a special type of an auction, which is called the second-price auction. As in regular auction n bidders place a bid which is price a bidder ready to pay. The auction is closed, that is, each bidder secretly informs the organizer of the auction price he is willing to pay. After that, the auction winner is the participant who offered the highest price. However, he pay not the price he offers, but the highest price among the offers of other participants (hence the name: the second-price auction).

Write a program that reads prices offered by bidders and finds the winner and the price he will pay. Consider that all of the offered prices are different.

Input

The first line of the input contains n $(2 \le n \le 1000)$ — number of bidders. The second line contains n distinct integer numbers $p_1, p_2, \ldots p_n$, separated by single spaces $(1 \le p_i \le 10000)$, where p_i stands for the price offered by the i-th bidder.

Output

The single output line should contain two integers: index of the winner and the price he will pay. Indices are 1-based.

Sample test(s)

input
2 5 7
output
2 5

input
3 10 2 8
output
1 8

input
6 3 8 2 9 4 14
output
6 9

B. Fly, freebies, fly!

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Everyone loves a freebie. Especially students.

It is well-known that if in the night before exam a student opens window, opens the student's record-book and shouts loudly three times "Fly, freebie, fly!" — then flown freebie helps him to pass the upcoming exam.

In the night before the exam on mathematical analysis n students living in dormitory shouted treasured words. The i-th student made a sacrament at the time t_i, where t_i is the number of seconds elapsed since the beginning of the night.

It is known that the freebie is a capricious and willful lady. That night the freebie was near dormitory only for T seconds. Therefore, if for two students their sacrament times differ for more than T, then the freebie didn't visit at least one of them.

Since all students are optimists, they really want to know what is the maximal number of students visited by the freebie can be.

Input

The first line of the input contains integer n ($1 \le n \le 100$), where n — the number of students shouted "Fly, freebie, fly!" The second line contains n positive integers t_i ($1 \le t_i \le 1000$).

The last line contains integer T ($1 \le T \le 1000$) — the time interval during which the freebie was near the dormitory.

Output

Print a single integer — the largest number of people who will pass exam tomorrow because of the freebie visit.

Sample test(s)

input
6 4 1 7 8 3 8 1
output
3

C. Diverse Substrings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

String diversity is the number of symbols that occur in the string at least once. Diversity of s will be denoted by $d(s)$. For example , d("aaa")=1, d("abacaba")=3.

Given a string s, consisting of lowercase Latin letters. Consider all its substrings. Obviously, any substring diversity is a number from 1 to $d(s)$. Find statistics about substrings diversity: for each k from 1 to $d(s)$, find how many substrings of s has a diversity of exactly k.

Input

The input consists of a single line containing s. It contains only lowercase Latin letters, the length of s is from 1 to $3 \cdot 10^5$.

Output

Print to the first line the value $d(s)$. Print sequence $t_1, t_2, ..., t_{d(s)}$ to the following lines, where t_i is the number of substrings of s having diversity of exactly i.

Sample test(s)

input
abca

output
3 4 3 3

input
aabacaabbad

output
4 14 19 28 5

Note

Consider the first example.

We denote by $s(i, j)$ a substring of "abca" with the indices in the segment $[i, j]$.

- $s(1, 1) =$ "a", d("a") $= 1$
- $s(2, 2) =$ "b", d("b") $= 1$
- $s(3, 3) =$ "c", d("c") $= 1$
- $s(4, 4) =$ "a", d("a") $= 1$
- $s(1, 2) =$ "ab", d("ab") $= 2$
- $s(2, 3) =$ "bc", d("bc") $= 2$
- $s(3, 4) =$ "ca", d("ca") $= 2$
- $s(1, 3) =$ "abc", d("abc") $= 3$
- $s(2, 4) =$ "bca", d("bca") $= 3$
- $s(1, 4) =$ "abca", d("abca") $= 3$

Total number of substring with diversity 1 is 4, with diversity 2 equals 3, 3 diversity is 3.

D. Game with Points

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

You are playing the following game. There are n points on a plane. They are the vertices of a regular n-polygon. Points are labeled with integer numbers from 1 to n. Each pair of distinct points is connected by a diagonal, which is colored in one of 26 colors. Points are denoted by lowercase English letters. There are three stones positioned on three distinct vertices. All stones are the same. With one move you can move the stone to another free vertex along some diagonal. The color of this diagonal must be the same as the color of the diagonal, connecting another two stones.

Your goal is to move stones in such way that the only vertices occupied by stones are 1, 2 and 3. You must achieve such position using minimal number of moves. Write a program which plays this game in an optimal way.

Input

In the first line there is one integer n ($3 \le n \le 70$) — the number of points. In the second line there are three space-separated integer from 1 to n — numbers of vertices, where stones are initially located.

Each of the following n lines contains n symbols — the matrix denoting the colors of the diagonals. Colors are denoted by lowercase English letters. The symbol j of line i denotes the color of diagonal between points i and j. Matrix is symmetric, so j-th symbol of i-th line is equal to i-th symbol of j-th line. Main diagonal is filled with '*' symbols because there is no diagonal, connecting point to itself.

Output

If there is no way to put stones on vertices 1, 2 and 3, print -1 on a single line. Otherwise, on the first line print minimal required number of moves and in the next lines print the description of each move, one move per line. To describe a move print two integers. The point from which to remove the stone, and the point to which move the stone. If there are several optimal solutions, print any of them.

Sample test(s)

input
4 2 3 4 *aba a*ab ba*b abb*
output
1 4 1

input
4 2 3 4 *abc a*ab ba*b cbb*
output
-1

Note

In the first example we can move stone from point 4 to point 1 because this points are connected by the diagonal of color 'a' and the diagonal connection point 2 and 3, where the other stones are located, are connected by the diagonal of the same color. After that stones will be on the points 1, 2 and 3.