Codeforces Round #155 (Div. 2)

A. Cards with Numbers

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

Petya has got $2n$ cards, each card contains some integer. The numbers on the cards can be the same. Let's index all cards by consecutive integers from 1 to $2n$. We'll denote the number that is written on a card with number i, as a_i. In order to play one entertaining game with his friends, Petya needs to split the cards into pairs so that each pair had equal numbers on the cards. Help Petya do that.

Input

The first line contains integer n ($1 \le n \le 3 \cdot 10^5$). The second line contains the sequence of $2n$ positive integers $a_1, a_2, ..., a_{2n}$ ($1 \le a_i \le 5000$) — the numbers that are written on the cards. The numbers on the line are separated by single spaces.

Output

If it is impossible to divide the cards into pairs so that cards in each pair had the same numbers, print on a single line integer `-1`. But if the required partition exists, then print n pairs of integers, a pair per line — the indices of the cards that form the pairs.

Separate the numbers on the lines by spaces. You can print the pairs and the numbers in the pairs in any order. If there are multiple solutions, print any of them.

Sample test(s)

input
3 20 30 10 30 20 10
output
4 2 1 5 6 3

input
1 1 2
output
-1

B. Jury Size

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

In 2013, the writers of Berland State University should prepare problems for n Olympiads. We will assume that the Olympiads are numbered with consecutive integers from 1 to n. For each Olympiad we know how many members of the jury must be involved in its preparation, as well as the time required to prepare the problems for her. Namely, the Olympiad number i should be prepared by p_i people for t_i days, the preparation for the Olympiad should be a continuous period of time and end exactly one day before the Olympiad. On the day of the Olympiad the juries who have prepared it, already do not work on it.

For example, if the Olympiad is held on December 9th and the preparation takes 7 people and 6 days, all seven members of the jury will work on the problems of the Olympiad from December, 3rd to December, 8th (the jury members won't be working on the problems of this Olympiad on December 9th, that is, some of them can start preparing problems for some other Olympiad). And if the Olympiad is held on November 3rd and requires 5 days of training, the members of the jury will work from October 29th to November 2nd.

In order not to overload the jury the following rule was introduced: one member of the jury can not work on the same day on the tasks for different Olympiads. Write a program that determines what the minimum number of people must be part of the jury so that all Olympiads could be prepared in time.

Input

The first line contains integer n — the number of Olympiads in 2013 ($1 \le n \le 100$). Each of the following n lines contains four integers m_i, d_i, p_i and t_i — the month and day of the Olympiad (given without leading zeroes), the needed number of the jury members and the time needed to prepare the i-th Olympiad ($1 \le m_i \le 12$, $d_i \ge 1$, $1 \le p_i, t_i \le 100$), d_i doesn't exceed the number of days in month m_i. The Olympiads are given in the arbitrary order. Several Olympiads can take place in one day.

Use the modern (Gregorian) calendar in the solution. Note that all dates are given in the year 2013. This is not a leap year, so February has 28 days.
Please note, the preparation of some Olympiad can start in 2012 year.

Output

Print a single number — the minimum jury size.

Sample test(s)

input
2 5 23 1 2 3 13 2 3
output
2

input
3 12 9 2 1 12 8 1 3 12 8 2 2
output
3

input
1 1 10 1 13
output
1

C. Anagram

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

String x is an *anagram* of string y, if we can rearrange the letters in string x and get exact string y. For example, strings "DOG" and "GOD" are anagrams, so are strings "BABA" and "AABB", but strings "ABBAC" and "CAABA" are not.

You are given two strings s and t of the same length, consisting of uppercase English letters. You need to get the anagram of string t from string s. You are permitted to perform the replacing operation: every operation is replacing some character from the string s by any other character. Get the anagram of string t in the least number of replacing operations. If you can get multiple anagrams of string t in the least number of operations, get the lexicographically minimal one.

The lexicographic order of strings is the familiar to us "dictionary" order. Formally, the string p of length n is lexicographically smaller than string q of the same length, if $p_1 = q_1, p_2 = q_2, ..., p_{k-1} = q_{k-1}, p_k < q_k$ for some k $(1 \le k \le n)$. Here characters in the strings are numbered from 1. The characters of the strings are compared in the alphabetic order.

Input

The input consists of two lines. The first line contains string s, the second line contains string t. The strings have the same length (from 1 to 10^5 characters) and consist of uppercase English letters.

Output

In the first line print z — the minimum number of replacement operations, needed to get an anagram of string t from string s. In the second line print the lexicographically minimum anagram that could be obtained in z operations.

Sample test(s)

input
ABA CBA
output
1 ABC

input
CDBABC ADCABD
output
2 ADBADC

Note

The second sample has eight anagrams of string t, that can be obtained from string s by replacing exactly two letters: "ADBADC", "ADDABC", "CDAABD", "CDBAAD", "CDBADA", "CDDABA", "DDAABC", "DDBAAC". These anagrams are listed in the lexicographical order. The lexicographically minimum anagram is "ADBADC".

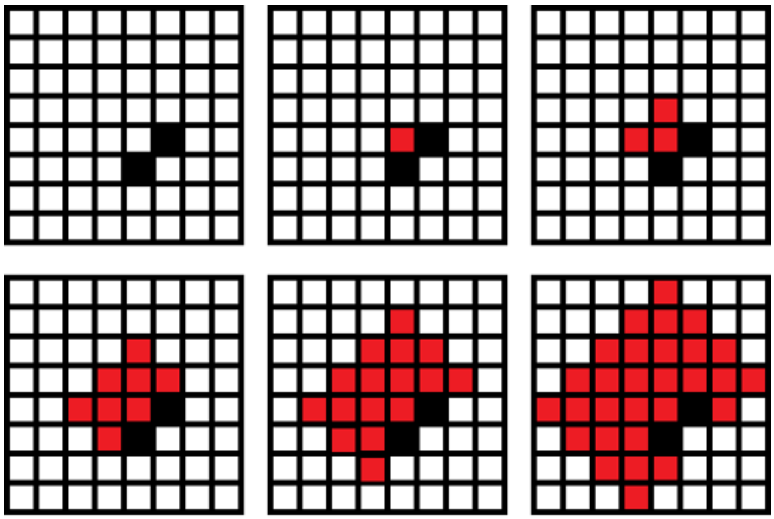
D. Rats

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

Rats have bred to hundreds and hundreds in the basement of the store, owned by Vasily Petrovich. Vasily Petrovich may have not noticed their presence, but they got into the habit of sneaking into the warehouse and stealing food from there. Vasily Petrovich cannot put up with it anymore, he has to destroy the rats in the basement. Since mousetraps are outdated and do not help, and rat poison can poison inattentive people as well as rats, he chose a radical way: to blow up two grenades in the basement (he does not have more).

In this problem, we will present the shop basement as a rectangular table of $n \times m$ cells. Some of the cells are occupied by walls, and the rest of them are empty. Vasily has been watching the rats and he found out that at a certain time they go to sleep, and all the time they sleep in the same places. He wants to blow up a grenade when this convenient time comes. On the plan of his basement, he marked cells with sleeping rats in them. Naturally, these cells are not occupied by walls.

Grenades can only blow up in a cell that is not occupied by a wall. The blast wave from a grenade distributes as follows. We assume that the grenade blast occurs at time 0. During this initial time only the cell where the grenade blew up gets 'clear'. If at time t some cell is clear, then at time $t+1$ those side-neighbouring cells which are not occupied by the walls get clear too (some of them could have been cleared before). The blast wave distributes for exactly d seconds, then it dies immediately.



An example of a distributing blast wave: Picture 1 shows the situation before the blast, and the following pictures show "clear" cells by time 0,1,2,3 and 4. Thus, the blast wave on the picture distributes for $d = 4$ seconds.

Vasily Petrovich wonders, whether he can choose two cells to blast the grenades so as to clear all cells with sleeping rats. Write the program that finds it out.

Input

The first line contains three integers n, m and d, separated by single spaces ($4 \le n, m \le 1000$, $1 \le d \le 8$). Next n lines contain the table that represents the basement plan. Each row of the table consists of m characters. Character "`X`" means that the corresponding cell is occupied by the wall, character "`.`" represents a empty cell, character "`R`" represents a empty cell with sleeping rats.

It is guaranteed that the first and the last row, as well as the first and the last column consist of characters "`X`". The plan has at least two empty cells. There is at least one cell with sleeping rats.

Output

If it is impossible to blow up all cells with sleeping rats, print a single integer `-1`. Otherwise, print four space-separated integers r_1, c_1, r_2, c_2, that mean that one grenade should go off in cell (r_1, c_1), and the other one — in cell (r_2, c_2).

Consider the table rows numbered from top to bottom from 1 to n and the table columns — from left to right from 1 to m. As r_1 and r_2 represent the row numbers, and c_1 and c_2 represent the column numbers in the table, they should fit the limits: $1 \le r_1, r_2 \le n$, $1 \le c_1, c_2 \le m$. It is forbidden to blow a grenade twice in the same cell. The blast waves of the grenades can intersect. It is possible that one grenade blast destroys no rats, and the other one destroys all of them.

Sample test(s)

input
4 4 1 XXXX XR.X X.RX XXXX
output
2 2 2 3

input

```
9 14 5
XXXXXXXXXXXXXX
X....R...R...X
X..R.........X
X....RXR..R..X
X..R...X.....X
XR.R...X.....X
X....XXR.....X
X....R..R.R..X
XXXXXXXXXXXXXX
```

output

```
2 3 6 9
```

input

```
7 7 1
XXXXXXX
X.R.R.X
X.....X
X..X..X
X..R..X
X....RX
XXXXXXX
```

output

```
-1
```

E. Dormitory

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

Student Vasya came to study in Berland State University from the country, so he is living in a dormitory. A semester has n days, and in each of those days his parents send him some food. In the morning of the i-th day he receives a_i kilograms of food that can be eaten on that day and on the next one (then the food goes bad and becomes unfit for consumption).

Every day Vasya eats v kilograms of food. It is known that Vasya's parents do not allow him to starve, so there always is enough food for Vasya. Vasya has m friends who sometimes live with him. Let's index the friends from 1 to m. Friend number j lives with Vasya from day l_j to day r_j, inclusive. Also, the j-th friend requires f_j kilograms of food per day. Usually Vasya's friends eat in the canteen, but sometimes generous Vasya feeds some of them. Every day Vasya can feed some friends who live with him this day (or may feed nobody).

Every time Vasya feeds his friend, he gives him as much food as the friend needs for the day, and Vasya's popularity rating at the University increases by one. Vasya cannot feed the same friend multiple times in one day. In addition, he knows that eating habits must be regular, so he always eats v kilograms of food per day.

Vasya wants so choose whom he will feed each day of the semester to make his rating as high as possible. Originally Vasya's rating is 0 because he is a freshman.

Input

The first line contains two integers n and v ($1 \le n, v \le 400$). The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 400$), separated by single spaces. Value a_i means that in the morning of the i-th day a_i kilograms of food come, the food is good for eating on day i and/or on day $i + 1$ (then the food goes bad). It is guaranteed that if Vasya doesn't feed anyone, there is a way for him to eat so as to consume v kilograms of food every day.

The third line contains integer m ($1 \le m \le 400$). Each of the following m lines describes one Vasya's friend: the j-th of these lines contains three integers l_j, r_j, f_j ($1 \le l_j \le r_j \le n, 1 \le f_j \le 400$), separated by single spaces.

Output

In the first line print the highest rating Vasya can reach. In the next n lines print, which friends Vasya needs to feed on each day. In the i-th of these lines first print the number of friends to feed on the i-th day, and then list the indexes of these friends. Print the friends in these lists in any order. If there are multiple optimal solutions, print any of them.

Sample test(s)

input

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

output

```
7
1 2
1 2
3 2 1 3
2 2 3
```