15-295 Fall #3 (Selection Round 1)

A. Treeland

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Treeland is an ancient country located on the territory of the modern Berland many many years ago. It is a well-known fact that Treeland consisted of n cities connected with $n - 1$ bidirectional roads. It was possible to travel from any city to any other one along the roads. The cities were numbered from 1 to n.

Nowadays we do not know what cities were connected by roads directly, but recently archaeologists have found new information that can help to reconstruct the road network of Treeland.

They found an ancient library with the diaries of a well-known traveler Biklouho-Baclay. The diaries contain interesting information: for each city i there is a list of all cities in the order of non-decreasing distance from i. The distances are calculated assuming that each road has length 1, so a distance between cities is the minimum number of roads to travel between them.

Formally, for each city i there is a list $C_i = \langle c_{i,1}, c_{i,2}, \ldots, c_{i,n} \rangle$ containing a permutation of cities (numbers from 1 to n) in such order that $d(i, c_{i,j}) \le d(i, c_{i,j+1})$ for every $j = 1 \ldots n - 1$, where $d(x, y)$ is a distance between cities x and y. Obviously, $c_{i,1} = i$ for each i. The cities that are equidistant from i can be listed in any order.

Your task is to restore a possible road network consistent with the found lists.

Input

The input consists of several test cases. The first line contains integer number t $(1 \le t \le 1000)$ — the number of test cases.

Then t blocks follow, each describing a single test case. The first line of a block contains integer number n $(2 \le n \le 2000)$ — the number of cities. The following n lines contain lists C_i, one list per line.

It is guaranteed that the required road network exists for each test case. The sum of n over all test cases doesn't exceed 2000.

Output

For each test case print $n - 1$ lines describing roads. Each line should contain a pair of cities connected with a road. You may print the roads and cities in a pair in any order. Print a blank line after the output of a test case.

If there are many solutions, print any of them.

Sample test(s)

input

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

output

```
2 1



2 3
3 4
```

```
5 4
4 1


2 1
3 1
```

B. Ilya Muromets

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Силачом слыву недаром — семерых одним ударом!

From the Russian cartoon on the German fairy tale.

Ilya Muromets is a legendary bogatyr. Right now he is struggling against Zmej Gorynych, a dragon with n heads numbered from 1 to n from left to right.

Making one sweep of sword Ilya Muromets can cut at most k contiguous heads of Zmej Gorynych. Thereafter heads collapse getting rid of empty space between heads. So in a moment before the second sweep all the heads form a contiguous sequence again.

As we all know, dragons can breathe fire. And so does Zmej Gorynych. Each his head has a firepower. The firepower of the i-th head is f_i.

Ilya Muromets has time for at most two sword sweeps. The bogatyr wants to reduce dragon's firepower as much as possible. What is the maximum total firepower of heads which Ilya can cut with at most two sword sweeps?

Input

The first line contains a pair of integer numbers n and k ($1 \le n, k \le 2 \cdot 10^5$) — the number of Gorynych's heads and the maximum number of heads Ilya can cut with a single sword sweep. The second line contains the sequence of integer numbers $f_1, f_2, \ldots, f_n$ ($1 \le f_i \le 2000$), where f_i is the firepower of the i-th head.

Output

Print the required maximum total head firepower that Ilya can cut.

Sample test(s)

input

```
8 2
1 3 3 1 2 3 11 1
```

output

```
20
```

input

```
4 100
10 20 30 40
```

output

```
100
```

C. ATM withdrawal

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

Vinh works for an ATM machine manufacturing company. The basic functionality of an ATM machine is cash withdrawal. When a user requests a cash withdrawal of W VND (Vietnamese Dong), the ATM has to dispense N money notes such that they sum up to W. For the next generation of ATM machine, Vinh is working on an algorithm to minimize the number N of money notes for each cash withdrawal transaction.

Your task is to help Vinh to do his job given that the money notes come in the values of $1000, 2000, 3000, 5000, 1000 * 10^1, 2000 * 10^1, 3000 * 10^1, 5000 * 10^1, ..., 1000 * 10^c, 2000 * 10^c, 3000 * 10^c, 5000 * 10^c$ where c is a positive integer and Vinh has unlimited supply of money notes for each value.

Input

The input file consists of several datasets. The first line of the input file contains the number of datasets which is a positive integer and is not greater than 1000. The following lines describe the datasets.

- The first line consists of one positive integer W ($W \leq 10^{18}$);
- The second line consists of one positive integer c ($c \leq 15$).

Output

For each dataset, write in one line two space-separated integers N and S where S is the number of ways to dispense the fewest number N of money notes. In case there is no way to serve the cash withdrawal request, write out 0 in one line instead.

Sample test(s)

input

```
2
1000
1
7000
1
```

output

```
1 1
2 1
```

D. Count Ways

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

You have a N x M matrix of cells. The cell $(i,\ j)$ represents the cell at row i and column j. You can go from one cell $(i,\ j)$ only down or right, that is to cells $(i\ +\ 1,\ j)$ or $(i,\ j\ +\ 1)$.

But K cells are blocked in the matrix(you can't visit a blocked cell). You are given coordinates of all these cells.

You have to count number of ways to reach cell (N, M) if you begin at cell $(1, 1)$.

Two ways are same if and only if path taken is exactly same in both ways.

Input

First line consists of T, denoting the number of test cases. Each test case consists of N, M and K in single line. Each of the next K lines contains two space separated integers (x_i, y_i), denoting the blocked cell's coordinates.

Output

For each test case print the required answer modulo $10^9 + 7$ in one line.

Constraints

- $1 \leq T \leq 10$
- $1 \leq N, M \leq 10^5$
- $0 \leq K \leq 10^3$
- $1 \leq x_i \leq N$
- $1 \leq y_i \leq M$

Sample test(s)

input

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

output

```
3
1
```

Note

Example test case 1:

No cell is blocked. 3 distinct paths are:

- $(1, 1)$ to $(1, 2)$ to $(2, 2)$ to $(3, 2)$.
- $(1, 1)$ to $(2, 1)$ to $(2, 2)$ to $(3, 2)$.
- $(1, 1)$ to $(2, 1)$ to $(3, 1)$ to $(3, 2)$.

Example test case 2: Only one valid path:

- $(1, 1)$ to $(2, 1)$ to $(2, 2)$ to $(2, 3)$.

E. FacePalm Accounting

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

An owner of a small company FacePalm has recently learned that the city authorities plan to offer to small businesses to participate in improving parks and garden squares. However, credible sources informed the FacePalm owner that the loss-making companies will not get such an offer. Moreover, the sources have also told how loss-making companies will be determined.

A company will be considered *loss-making* if for every k contiguous days the total income of the company is negative.

The FacePalm owner discussed the situation with his chief accountant, and they decided to change the report so that the company would look loss-making.

The company report for n days can be represented as a sequence of integers $a_1, a_2, ..., a_n$, where a_i is the company income in the day i (negative values correspond to losses).

The accountant can change any values in this sequence, but no updated value can become less than the smallest value in the original report — otherwise the updated report will look absolutely implausible. Besides, the accountant wants the total change of the values to be as small as possible.

We will assume that the total change of the values is $\sum_{i=1}^{n} |\tilde{a}_i - a_i|$, where $\tilde{a}_i$ is the i-th value in the updated report.

Your task is to calculate the minimum required total change of the values and provide the updated report.

Input

The first line contains integers n and k $(1 \le k \le n \le 2 \cdot 10^5)$ — the number of days in the report and the number of days in the definition of loss-making company.

The second line contains n space-separated integers $a_1, a_2, ..., a_n$ $(- 10000 \le a_i \le 10000)$, where a_i is the company income in day i.

It is guaranteed that at least one value of a_i is negative.

Output

In the first line print the required minimum total change. In the second line print the corresponding updated report. No value in the updated report can be less than the minimum value of the original report.

If there are multiple solutions, print any of them.

Sample test(s)

input

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

output

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

input

```
8 3
2 1 -3 -2 4 -3 0 2
```

output

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

input

```
4 2
-2 1 -2 1
```

output

```
0
-2 1 -2 1
```

input

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

output

```
12
-5 -5 9
```

F. Colored Blankets

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Recently Polycarp has opened a blankets store and he faced with many challenges.

He has got k blankets. A blanket has two sides, each of k blankets has at most one colored side. So either both sides are uncolored or one side is colored and the other one is not. If a side is colored, one of n possible colors is used. It is known that k is divisible by n.

Polycarp wants to color all uncolored sides in such a way that:

- each blanket will have both sides colored (one color per side), the colors can be the same or different, all the used colors are from n possible colors;
- it will be possible to compose n kits with $\frac{k}{n}$ blankets in each kit that blankets in each kit are *identically colored*.

It is allowed to turn over blankets to determine that they are *identically colored*: for example, red-blue and blue-red blankets are *identically colored*. Blankets in different kits can be *identically colored*.

Input

The first line contains two integer numbers k and n $(1 \le n \le k \le 1000)$ — number of blankets and colors. It is guaranteed that k is divisible by n. The second line contains a sequence of integers $c_1, c_2, ..., c_k$ $(1 \le c_i \le n$ or $c_i = -1)$, where c_i stands for the color of the colored side of the i-th blanket or -1 if it is uncolored.

Output

If there is no solution for the problem, print "No" in the first line. Otherwise print a line containing "Yes" and k lines describing each blanket. The i-th line should contain a pair of colors (integers in the range $1, 2, ..., n$) used for the i-th blanket. You may print colors in pairs in any order.

If there are multiple solutions, print any of them.

Sample test(s)

input

```
6 2
1 1 2 2 -1 2
```

output

```
Yes
1 2
2 1
2 2
2 2
2 1
2 2
```

input

```
8 4
4 -1 1 -1 4 3 -1 -1
```

output

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

G. Laughing Out Loud

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Little Toojee was a happy go lucky boy. He seemed to find most, if not all, things funny. One day he read a word and started laughing a lot. Turns out that the word consisted only of the letters L and O. Whenever he saw the subsequence 'LOL' in the word, he laughed for 1 second. Given t strings, find out for how long Toojee laughed on seeing each string.

Input
The first line contains t queries. This is followed by t lines each containing one string S. String S consists only of capital alphabets.

Output
Output for each string on a new line.

Constraints

- $1 \leq t \leq 10$
- $1 \leq |S| \leq 10^5$

Sample test(s)

input
```
2
LOL
LOLOL
```
output
```
1
4
```

Note
$Test1$: On observation, we can tell that there is only 1 occurrence of LOL.

$Test2$: Let the string be 0-indexed and let $V = \{a, b, c\}$ denote the indices that make up the string "LOL", where a is index of the 1st 'L', b is index of the 'O' and c is the index of the 2nd 'L'. So, V can be $\{0, 1, 2\}$, $\{2, 3, 4\}$, $\{0, 1, 4\}$ and $\{0, 3, 4\}$. We see that there are 4 occurrences of the string "LOL".

Problem H. Minimal Agapov Code

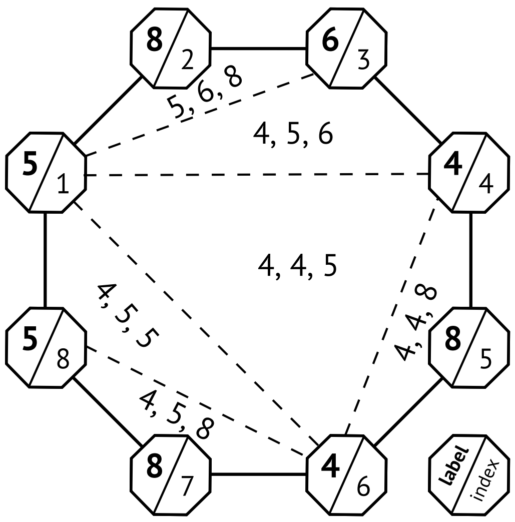
Input file:	`stdin`
Output file:	`stdout`
Time limit:	8 seconds
Memory limit:	512 megabytes

Triangulation is a decomposition of a polygon into a set of triangles that the triangle vertices are in the vertices of the polygon. The triangles in a triangulation can't have internal common points, and they should cover all the polygon. It is easy to see that triangulation of n-gon always consists of $n-2$ triangles.

You are given a regular polygon with n vertices. The vertices are labeled with integer numbers. *Agapov code* of its triangulation is constructed as follows:

- write a code for each triangle: a sequence of labels of a triangle vertices in non-decreasing order;
- concatenate all triangle codes in such order that the resulting sequence is lexicographically minimal.

A sequence $a = \langle a_1, a_2, \ldots, a_m\rangle$ is lexicographically less than $b = \langle b_1, b_2, \ldots, b_m\rangle$ if there is such index i $(1 \le i \le m)$ that $a_i < b_i$ and $a_j = b_j$ for all j $(1 \le j < i)$.



Triangle codes are written in triangles, the *Agapov code* is $\langle 4,4,5,4,4,8,4,5,5,4,5,6,4,5,8,5,6,8\rangle$.

Given a labeled regular polygon write a program to find the triangulation with lexicographically minimal *Agapov code*.

Input

The first line contains integer n $(3 \le n \le 5 \cdot 10^5)$. The second line contains n space-separated integers: $c_1, c_2, \ldots, c_n$ $(1 \le c_i \le n)$, where c_i denotes the label on the i-th vertex. Vertices are indexed clockwise.

Output

Print $n-2$ lines describing the required triangulation, one line per triangle. In each line print three integers: indices of triangle vertices in order of non-decreasing of their labels. Lines should be ordered in such a way that concatenation of triangle codes in order of output will give the required lexicographically minimal *Agapov code*.

If there are multiple solutions, print any of them.

Examples

stdin	stdout

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

I. Election of a Mayor

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The Berland capital is preparing for mayoral election. There are two candidates for the position: the current mayor and his rival. The rival is a serious competitor, and it's not easy for current mayor to win the election.

The candidate will be declared the winner if he wins at *more* than a half of *all* polling stations. The results of the polling stations are supplied independently. The station winner is the candidate who gets *more* than a half of the votes of this station. No candidate is declared a winner of a polling station if both candidates have got the same number of votes at this station. Similarly, no candidate is declared a winner of the election if both candidates won at the same number of polling stations.

All eligible voters are going to take part in the election and have already decided whom to give their vote to. The campaign headquarters of the current mayor has collected data from all n stations of the city, and now for every station it is known how many people will vote for the current mayor and how many people will vote for his opponent.

The results have been disappointing for the current mayor, but his staff came up with a way to win the election. It was suggested to merge some pairs of polling stations in such a way that the current mayor will become the election winner. However, (for the sake of credibility), the proposed plan must comply with two conditions. Firstly, it is possible to merge only the pairs of stations with adjacent numbers (i.e., station j may be merged with either station $j - 1$, or with station $j + 1$). The resulting station cannot be merged again. Secondly, the number of such mergers for obvious reasons must be as few as possible.

Your task is to help the current mayor's campaign headquarters and produce such plan.

Input

The first line contains single integer n $(2 \le n \le 2 \cdot 10^5)$ — the number of the polling stations.

Each of the following n lines contains two integers m_j and r_j $(0 \le m_j, r_j \le 10^5)$ — the number of voters at station j who plan to vote for the current mayor and his rival correspondingly.

Output

If current mayor cannot win the election at any condition, print a single number -1 to the first line.

Otherwise, to the first line print an integer u — the minimum number of polling station mergers the current mayor needs to perform in order to win. To each of the next u lines print a pair of integers — the numbers of the merged stations. The pairs as well as the numbers within each pair can be printed in any order. If there are multiple solutions, print any of them.

Sample test(s)

input

```
7
15 8
8 10
14 14
12 13
13 12
21 10
20 30
```

output

```
2
1 2
6 7
```

input

```
2
1 5
5 1
```

output

```
-1
```

input

```
2
10 9
15 7
```

output

```
0
```