Codeforces Round #375 (Div. 2)

A. The New Year: Meeting Friends

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

There are three friend living on the straight line Ox in Lineland. The first friend lives at the point x_1, the second friend lives at the point x_2, and the third friend lives at the point x_3. They plan to celebrate the New Year together, so they need to meet at one point. What is the minimum total distance they have to travel in order to meet at some point and celebrate the New Year?

It's guaranteed that the optimal answer is always integer.

Input

The first line of the input contains three **distinct** integers x_1, x_2 and x_3 $(1 \le x_1, x_2, x_3 \le 100)$ — the coordinates of the houses of the first, the second and the third friends respectively.

Output

Print one integer — the minimum total distance the friends need to travel in order to meet together.

Examples

input
7 1 4
output
6

input
30 20 10
output
20

Note

In the first sample, friends should meet at the point 4. Thus, the first friend has to travel the distance of 3 (from the point 7 to the point 4), the second friend also has to travel the distance of 3 (from the point 1 to the point 4), while the third friend should not go anywhere because he lives at the point 4.

B. Text Document Analysis

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

Modern text editors usually show some information regarding the document being edited. For example, the number of words, the number of pages, or the number of characters.

In this problem you should implement the similar functionality.

You are given a string which only consists of:

- uppercase and lowercase English letters,
- underscore symbols (they are used as separators),
- parentheses (both opening and closing).

It is guaranteed that each opening parenthesis has a succeeding closing parenthesis. Similarly, each closing parentheses has a preceding opening parentheses matching it. For each pair of matching parentheses there are no other parenthesis between them. In other words, each parenthesis in the string belongs to a matching "opening-closing" pair, and such pairs can't be nested.

For example, the following string is valid: "`_Hello_Vasya(and_Petya)__bye_(and_OK)`".

Word is a maximal sequence of consecutive letters, i.e. such sequence that the first character to the left and the first character to the right of it is an underscore, a parenthesis, or it just does not exist. For example, the string above consists of seven words: "`Hello`", "`Vasya`", "`and`", "`Petya`", "`bye`", "`and`" and "`OK`". Write a program that finds:

- the length of the longest word outside the parentheses (print `0`, if there is no word outside the parentheses),
- the number of words inside the parentheses (print `0`, if there is no word inside the parentheses).

Input

The first line of the input contains a single integer n ($1 \le n \le 255$) — the length of the given string. The second line contains the string consisting of only lowercase and uppercase English letters, parentheses and underscore symbols.

Output

Print two space-separated integers:

- the length of the longest word outside the parentheses (print `0`, if there is no word outside the parentheses),
- the number of words inside the parentheses (print `0`, if there is no word inside the parentheses).

Examples

input
37 _Hello_Vasya(and_Petya)__bye_(and_OK)
output
5 4

input
37 _a_(_b___c)__de_f(g_)__h__i(j_k_l)m__
output
2 6

input
27 (LoooonG)__shOrt__(LoooonG)
output
5 2

input
5 (___)
output

```
0 0
```

Note

In the first sample, the words "Hello", "Vasya" and "bye" are outside any of the parentheses, and the words "and", "Petya", "and" and "OK" are inside. Note, that the word "and" is given twice and you should count it twice in the answer.

C. Polycarp at the Radio

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

Polycarp is a music editor at the radio station. He received a playlist for tomorrow, that can be represented as a sequence $a_1, a_2, ..., a_n$, where a_i is a band, which performs the i-th song. Polycarp likes bands with the numbers from 1 to m, but he doesn't really like others.

We define as b_j the number of songs the group j is going to perform tomorrow. Polycarp wants to change the playlist in such a way that the minimum among the numbers $b_1, b_2, ..., b_m$ will be as large as possible.

Find this maximum possible value of the minimum among the b_j ($1 \le j \le m$), and the minimum number of changes in the playlist Polycarp needs to make to achieve it. One change in the playlist is a replacement of the performer of the i-th song with any other group.

Input

The first line of the input contains two integers n and m ($1 \le m \le n \le 2000$).

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$), where a_i is the performer of the i-th song.

Output

In the first line print two integers: the maximum possible value of the minimum among the b_j ($1 \le j \le m$), where b_j is the number of songs in the changed playlist performed by the j-th band, and the minimum number of changes in the playlist Polycarp needs to make.

In the second line print the changed playlist.

If there are multiple answers, print any of them.

Examples

input
4 2 1 2 3 2
output
2 1 1 2 1 2

input
7 3 1 3 2 2 2 2 1
output
2 1 1 3 3 2 2 2 1

input
4 4 1000000000 100 7 1000000000
output
1 4 1 2 3 4

Note

In the first sample, after Polycarp's changes the first band performs two songs ($b_1 = 2$), and the second band also performs two songs ($b_2 = 2$). Thus, the minimum of these values equals to 2. It is impossible to achieve a higher minimum value by any changes in the playlist.

In the second sample, after Polycarp's changes the first band performs two songs ($b_1 = 2$), the second band performs three songs ($b_2 = 3$), and the third band also performs two songs ($b_3 = 2$). Thus, the best minimum value is 2.

D. Lakes in Berland

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

The map of Berland is a rectangle of the size $n \times m$, which consists of cells of size 1×1. Each cell is either land or water. The map is surrounded by the ocean.

Lakes are the maximal regions of water cells, connected by sides, which are not connected with the ocean. Formally, lake is a set of water cells, such that it's possible to get from any cell of the set to any other without leaving the set and moving only to cells adjacent by the side, none of them is located on the border of the rectangle, and it's impossible to add one more water cell to the set such that it will be connected with any other cell.

You task is to fill up with the earth the minimum number of water cells so that there will be **exactly** k lakes in Berland. Note that the initial number of lakes on the map is **not less** than k.

Input

The first line of the input contains three integers n, m and k ($1 \le n, m \le 50$, $0 \le k \le 50$) — the sizes of the map and the number of lakes which should be left on the map.

The next n lines contain m characters each — the description of the map. Each of the characters is either '`.`' (it means that the corresponding cell is water) or '`*`' (it means that the corresponding cell is land).

It is guaranteed that the map contain at least k lakes.

Output

In the first line print the minimum number of cells which should be transformed from water to land.

In the next n lines print m symbols — the map after the changes. The format must strictly follow the format of the map in the input data (there is no need to print the size of the map). If there are several answers, print any of them.

It is guaranteed that the answer exists on the given data.

Examples

input

```
5 4 1
****
*..*
****
**.*
..**
```

output

```
1
****
*..*
****
****
..**
```

input

```
3 3 0
***
*.*
***
```

output

```
1
***
***
***
```

Note

In the first example there are only two lakes — the first consists of the cells $(2, 2)$ and $(2, 3)$, the second consists of the cell $(4, 3)$. It is profitable to cover the second lake because it is smaller. Pay attention that the area of water in the lower left corner is not a lake because this area share a border with the ocean.

E. One-Way Reform

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

There are n cities and m two-way roads in Berland, each road connects two cities. It is known that there is no more than one road connecting each pair of cities, and there is no road which connects the city with itself. It is possible that there is no way to get from one city to some other city using only these roads.

The road minister decided to make a reform in Berland and to orient all roads in the country, i.e. to make each road one-way. The minister wants to **maximize** the number of cities, for which the number of roads that begins in the city **equals** to the number of roads that ends in it.

Input

The first line contains a positive integer t ($1 \le t \le 200$) — the number of testsets in the input.

Each of the testsets is given in the following way. The first line contains two integers n and m ($1 \le n \le 200$, $0 \le m \le n \cdot (n - 1) / 2$) — the number of cities and the number of roads in Berland.

The next m lines contain the description of roads in Berland. Each line contains two integers u and v ($1 \le u, v \le n$) — the cities the corresponding road connects. It's guaranteed that there are no self-loops and multiple roads. It is possible that there is no way along roads between a pair of cities.

It is guaranteed that the total number of cities in all testset of input data doesn't exceed 200.

Pay attention that for **hacks**, you can only use tests consisting of **one testset**, so t should be equal to one.

Output

For each testset print the maximum number of such cities that the number of roads that begins in the city, is equal to the number of roads that ends in it.

In the next m lines print oriented roads. First print the number of the city where the road begins and then the number of the city where the road ends. If there are several answers, print any of them. It is allowed to print roads in each test in arbitrary order. Each road should be printed exactly once.

Example

input
2 5 5 2 1 4 5 2 3 1 3 3 5 7 2 3 7 4 2
output
3 1 3 3 5 5 4 3 2 2 1 3 2 4 3 7

F. st-Spanning Tree

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected connected graph consisting of n vertices and m edges. There are no loops and no multiple edges in the graph.

You are also given two distinct vertices s and t, and two values d_s and d_t. Your task is to build any spanning tree of the given graph (note that the graph is not weighted), such that the degree of the vertex s doesn't exceed d_s, and the degree of the vertex t doesn't exceed d_t, or determine, that there is no such spanning tree.

The *spanning tree* of the graph G is a subgraph which is a tree and contains all vertices of the graph G. In other words, it is a connected graph which contains n - 1 edges and can be obtained by removing some of the edges from G.

The degree of a vertex is the number of edges incident to this vertex.

Input

The first line of the input contains two integers n and m ($2 \le n \le 200\,000$, $1 \le m \le min(400\,000, n \cdot (n - 1) / 2)$) — the number of vertices and the number of edges in the graph.

The next m lines contain the descriptions of the graph's edges. Each of the lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) — the ends of the corresponding edge. It is guaranteed that the graph contains no loops and no multiple edges and that it is connected.

The last line contains four integers s, t, d_s, d_t ($1 \le s, t \le n$, $s \ne t$, $1 \le d_s, d_t \le n - 1$).

Output

If the answer doesn't exist print "No" (without quotes) in the only line of the output.

Otherwise, in the first line print "Yes" (without quotes). In the each of the next $(n - 1)$ lines print two integers — the description of the edges of the spanning tree. Each of the edges of the spanning tree must be printed exactly once.

You can output edges in any order. You can output the ends of each edge in any order.

If there are several solutions, print any of them.

Examples

input
3 3 1 2 2 3 3 1 1 2 1 1

output
Yes 3 2 1 3

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

output
Yes 1 3 5 7 3 2 7 4 2 4 6 1