A. Ivan the Fool and the Probability Theory

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

Recently Ivan the Fool decided to become smarter and study the probability theory. He thinks that he understands the subject fairly well, and so he began to behave like he already got PhD in that area.

To prove his skills, Ivan decided to demonstrate his friends a concept of random picture. A picture is a field of n rows and m columns, where each cell is either black or white. Ivan calls the picture random if for every cell it has **at most** one adjacent cell of the same color. Two cells are considered adjacent if they share a side.

Ivan's brothers spent some time trying to explain that it's not how the randomness usually works. Trying to convince Ivan, they want to count the number of different random (according to Ivan) pictures. Two pictures are considered different if at least one cell on those two picture is colored differently. Since the number of such pictures may be quite large, print it modulo $10^9 + 7$.

Input

The only line contains two integers n and m ($1 \le n, m \le 100\,000$), the number of rows and the number of columns of the field.

Output

Print one integer, the number of random pictures modulo $10^9 + 7$.

Example

input

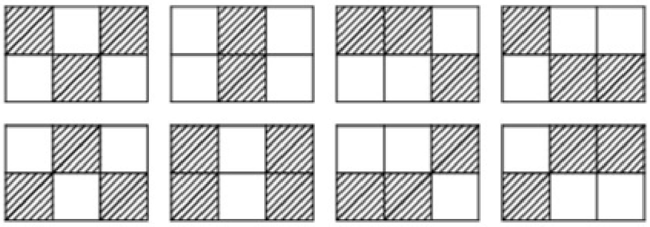
```
2 3
```

output

```
8
```

Note

The picture below shows all possible random pictures of size 2 by 3.



B. The World Is Just a Programming Task (Hard Version)

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

This is a harder version of the problem. In this version, $n \le 300\,000$.

Vasya is an experienced developer of programming competitions' problems. As all great minds at some time, Vasya faced a creative crisis. To improve the situation, Petya gifted him a string consisting of opening and closing brackets only. Petya believes, that the beauty of the bracket string is a number of its cyclical shifts, which form a correct bracket sequence.

To digress from his problems, Vasya decided to select two positions of the string (**not necessarily distinct**) and swap characters located at this positions with each other. Vasya will apply this operation exactly once. He is curious what is the maximum possible beauty he can achieve this way. Please help him.

We remind that bracket sequence s is called correct if:

- s is empty;
- s is equal to "(t)", where t is correct bracket sequence;
- s is equal to $t_1 t_2$, i.e. concatenation of t_1 and t_2, where t_1 and t_2 are correct bracket sequences.

For example, "(()())", "()" are correct, while ")(" and "())" are not.

The cyclical shift of the string s of length n by k ($0 \leq k < n$) is a string formed by a concatenation of the last k symbols of the string s with the first $n-k$ symbols of string s. For example, the cyclical shift of string "(())()" by 2 equals "()(())".

Cyclical shifts i and j are considered different, if $i \neq j$.

Input

The first line contains an integer n ($1 \leq n \leq 300\,000$), the length of the string.

The second line contains a string, consisting of exactly n characters, where each of the characters is either "(" or ")".

Output

The first line should contain a single integer — the largest beauty of the string, which can be achieved by swapping some two characters.

The second line should contain integers l and r ($1 \leq l, r \leq n$) — the indices of two characters, which should be swapped in order to maximize the string's beauty.

In case there are several possible swaps, print any of them.

Examples

input

```
10
()()())(()
```

output

```
5
8 7
```

input

```
12
)(()(()())()
```

output

```
4
5 10
```

input

```
6
)))(()
```

output

```
0
1 1
```

Note

In the first example, we can swap 7-th and 8-th character, obtaining a string "()()()()()". The cyclical shifts by $0, 2, 4, 6, 8$ of this string form a correct bracket sequence.

In the second example, after swapping 5-th and 10-th character, we obtain a string ")(())()()(()". The cyclical shifts by $11, 7, 5, 3$ of this string form a correct bracket sequence.

In the third example, swap of any two brackets results in 0 cyclical shifts being correct bracket sequences.

C. Queue in the Train

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

There are n seats in the train's car and there is exactly one passenger occupying every seat. The seats are numbered from 1 to n from left to right. The trip is long, so each passenger will become hungry at some moment of time and will go to take boiled water for his noodles. The person at seat i ($1 \leq i \leq n$) will decide to go for boiled water at minute t_i.

Tank with a boiled water is located to the left of the 1-st seat. In case too many passengers will go for boiled water simultaneously, they will form a queue, since there can be only one passenger using the tank at each particular moment of time. Each passenger uses the tank for exactly p minutes. We assume that the time it takes passengers to go from their seat to the tank is negligibly small.

Nobody likes to stand in a queue. So when the passenger occupying the i-th seat wants to go for a boiled water, he will first take a look on all seats from 1 to $i-1$. In case at least one of those seats is empty, he assumes that those people are standing in a queue right now, so he would be better seating for the time being. However, at the very first moment he observes that all seats with numbers smaller than i are busy, he will go to the tank.

There is an unspoken rule, that in case at some moment several people can go to the tank, than only the leftmost of them (that is, seating on the seat with smallest number) will go to the tank, while all others will wait for the next moment.

Your goal is to find for each passenger, when he will receive the boiled water for his noodles.

Input

The first line contains integers n and p ($1 \le n \le 100\,000$, $1 \le p \le 10^9$) — the number of people and the amount of time one person uses the tank.

The second line contains n integers $t_1, t_2, \dots, t_n$ ($0 \le t_i \le 10^9$) — the moments when the corresponding passenger will go for the boiled water.

Output

Print n integers, where i-th of them is the time moment the passenger on i-th seat will receive his boiled water.

Example

input

```
5 314
0 310 942 628 0
```

output

```
314 628 1256 942 1570
```

Note

Consider the example.

At the 0-th minute there were two passengers willing to go for a water, passenger 1 and 5, so the first passenger has gone first, and returned at the 314-th minute. At this moment the passenger 2 was already willing to go for the water, so the passenger 2 has gone next, and so on. In the end, 5-th passenger was last to receive the boiled water.

D. Catowice City

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

In the Catowice city next weekend the cat contest will be held. However, the jury members and the contestants haven't been selected yet. There are n residents and n cats in the Catowice, and each resident has exactly one cat living in his house. The residents and cats are numbered with integers from 1 to n, where the i-th cat is living in the house of i-th resident.

Each Catowice resident is in friendship with several cats, including the one living in his house. In order to conduct a contest, at least one jury member is needed and at least one cat contestant is needed. Of course, every jury member should know none of the contestants. For the contest to be successful, it's also needed that the number of jury members plus the number of contestants is equal to n.

Please help Catowice residents to select the jury and the contestants for the upcoming competition, or determine that it's impossible to do.

Input

The first line contains an integer t ($1 \le t \le 100\,000$), the number of test cases. Then description of t test cases follow, where each description is as follows:

The first line contains integers n and m ($1 \le n \le m \le 10^6$), the number of Catowice residents and the number of friendship pairs between residents and cats.

Each of the next m lines contains integers a_i and b_i ($1 \le a_i, b_i \le n$), denoting that a_i-th resident is acquaintances with b_i-th cat. It's guaranteed that each pair of some resident and some cat is listed at most once.

It's guaranteed, that for every i there exists a pair between i-th resident and i-th cat.

Different test cases are separated with an empty line.

It's guaranteed, that the sum of n over all test cases is at most 10^6 and that the sum of m over all test cases is at most 10^6.

Output

For every test case print:

- "No", if it's impossible to select the jury and contestants.
- Otherwise print "Yes".
  In the second line print two integers j and p ($1 \le j$, $1 \le p$, $j + p = n$) — the number of jury members and the number of contest participants.

  In the third line print j distinct integers from 1 to n, the indices of the residents forming a jury.

  In the fourth line print p distinct integers from 1 to n, the indices of the cats, which will participate in the contest.

  In case there are several correct answers, print any of them.

Example

input

```
4
```

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

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

1 1
1 1

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

output

```
Yes
2 1
1 3
2
Yes
1 2
2
1 3
No
No
```

Note

In the first test case, we can select the first and the third resident as a jury. Both of them are not acquaintances with a second cat, so we can select it as a contestant.

In the second test case, we can select the second resident as a jury. He is not an acquaintances with a first and a third cat, so they can be selected as contestants.

In the third test case, the only resident is acquaintances with the only cat, so they can't be in the contest together. So it's not possible to make a contest with at least one jury and at least one cat.

In the fourth test case, each resident is acquaintances with every cat, so it's again not possible to make a contest with at least one jury and at least one cat.

E. Turtle

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Kolya has a turtle and a field of size $2 \times n$. The field rows are numbered from 1 to 2 from top to bottom, while the columns are numbered from 1 to n from left to right.

Suppose in each cell of the field there is a lettuce leaf. The energy value of lettuce leaf in row i and column j is equal to $a_{i,j}$. The turtle is initially in the top left cell and wants to reach the bottom right cell. The turtle can only move right and down and among all possible ways it will choose a way, maximizing the total energy value of lettuce leaves (in case there are several such paths, it will choose any of them).

Kolya is afraid, that if turtle will eat too much lettuce, it can be bad for its health. So he wants to reorder lettuce leaves in the field, so that the energetic cost of leaves eaten by turtle will be minimized.

Input

The first line contains an integer n ($2 \le n \le 25$) — the length of the field.

The second line contains n integers $a_{1,i}$ ($0 \le a_{1,i} \le 50\,000$), the energetic cost of lettuce leaves in the first row of the field.

The third line contains n integers $a_{2,i}$ ($0 \le a_{2,i} \le 50\,000$), the energetic cost of lettuce leaves in the second row of the field.

Output

Print two lines with n integers in each — the optimal reordering of lettuce from the input data.

In case there are several optimal ways to reorder lettuce, print any of them.

Examples

input

```
2
1 4
```

```
2 3
```

output

```
1 3
4 2
```

input

```
3
0 0 0
0 0 0
```

output

```
0 0 0
0 0 0
```

input

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

output

```
0 0 1
0 1 0
```

Note

In the first example, after reordering, the turtle will eat lettuce with total energetic cost $1 + 4 + 2 = 7$.

In the second example, the turtle will eat lettuce with energetic cost equal 0.

In the third example, after reordering, the turtle will eat lettuce with total energetic cost equal 1.

F. Swiper, no swiping!

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

I'm the Map, I'm the Map! I'm the MAP!!!

Map

In anticipation of new adventures Boots wanted to do a good deed. After discussion with the Map and Backpack, they decided to gift Dora a connected graph. After a long search, Boots chose t graph's variants, which Dora might like. However fox Swiper wants to spoil his plan.

The Swiper knows, that Dora now is only able to count up to 3, so he has came up with a following idea. He wants to steal some non-empty set of vertices, so that the Dora won't notice the loss. He has decided to steal some non-empty set of vertices, so that after deletion of the stolen vertices and edges adjacent to them, every **remaining** vertex wouldn't change it's degree modulo 3. The degree of a vertex is the number of edges it is adjacent to. It would've been suspicious to steal all the vertices, so Swiper needs another plan.

Boots are sure, that the crime can not be allowed. However they are afraid, that they won't be able to handle this alone. So Boots decided to ask for your help. Please determine for every graph's variant whether the Swiper can perform the theft or not.

Input

The first line contains a single integer t ($1 \le t \le 100\,000$) — the number of graph variants.

The first line of each variant contains integers n, m ($1 \le n \le 500\,000$, $0 \le m \le 500\,000$), the number of vertexes and edges in the graph.

Then m lines follow, each containing integers a_i, b_i ($1 \le a_i, b_i \le n$), the indices of the vertices connected with a corresponding edge.

It's guaranteed, that the graph is connected and doesn't contain multiple edges or self-loops.

It's guaranteed, that the sum of n over all variants is at most $500\,000$ and that the sum of m over all variants is at most $500\,000$.

Descriptions of graph's variants are separated with an empty line.

Output

For each variant:

- In case the answer exists, print "Yes" and then the answer itself.
  The first line should contain an integer c ($1 < c < n$), the number of vertices the Crook can steal, without Dora noticing the loss. On the next line print c distinct integers, the indices of the graph's vertices in arbitrary order.
- Otherwise print "No".

In case there are several correct ways to steal the vertices, print any of them.

Please note, that it's not required to maximize the number of stolen vertices.

Example

input

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

6 6
1 2
1 3
2 3
2 5
2 6
2 4

8 12
1 2
1 3
2 3
1 4
4 5
5 1
3 6
3 7
3 8
6 1
7 1
8 1
```

output

```
No
Yes
3
4 5 6
Yes
3
6 7 8
```

Note

The picture below shows the third variant from the example test. The set of the vertices the Crook can steal is denoted with bold.