Codeforces Round #134 (Div. 2)

A. Mountain Scenery

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

Little Bolek has found a picture with n mountain peaks painted on it. The n painted peaks are represented by a non-closed polyline, consisting of $2n$ segments. The segments go through $2n + 1$ points with coordinates $(1, y_1)$, $(2, y_2)$, ..., $(2n + 1, y_{2n+1})$, with the i-th segment connecting the point (i, y_i) and the point $(i + 1, y_{i+1})$. For any even i $(2 \le i \le 2n)$ the following condition holds: $y_{i-1} < y_i$ and $y_i > y_{i+1}$.

We shall call a vertex of a polyline with an even x coordinate a *mountain peak*.

The figure to the left shows the initial picture, the figure to the right shows what the picture looks like after Bolek's actions. The affected peaks are marked red, $k = 2$.

Bolek fancied a little mischief. He chose exactly k mountain peaks, rubbed out the segments that went through those peaks and increased each peak's height by one (that is, he increased the y coordinate of the corresponding points). Then he painted the missing segments to get a new picture of mountain peaks. Let us denote the points through which the new polyline passes on Bolek's new picture as $(1, r_1)$, $(2, r_2)$, ..., $(2n + 1, r_{2n+1})$.

Given Bolek's final picture, restore the initial one.

Input

The first line contains two space-separated integers n and k $(1 \le k \le n \le 100)$. The next line contains $2n + 1$ space-separated integers $r_1, r_2, \ldots, r_{2n+1}$ $(0 \le r_i \le 100)$ — the y coordinates of the polyline vertices on Bolek's picture.

It is guaranteed that we can obtain the given picture after performing the described actions on some picture of mountain peaks.

Output

Print $2n + 1$ integers $y_1, y_2, \ldots, y_{2n+1}$ — the y coordinates of the vertices of the polyline on the initial picture. If there are multiple answers, output any one of them.

Examples

input
3 2 0 5 3 5 1 5 2
output
0 5 3 4 1 4 2

input
1 1 0 2 0
output
0 1 0

B. Airport

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

Lolek and Bolek are about to travel abroad by plane. The local airport has a special "Choose Your Plane" offer. The offer's conditions are as follows:

- it is up to a passenger to choose a plane to fly on;
- if the chosen plane has x ($x > 0$) empty seats at the given moment, then the ticket for such a plane costs x zlotys (units of Polish currency).

The only ticket office of the airport already has a queue of n passengers in front of it. Lolek and Bolek have not stood in the queue yet, but they are already wondering what is the maximum and the minimum number of zlotys the airport administration can earn if all n passengers buy tickets according to the conditions of this offer?

The passengers buy tickets in turn, the first person in the queue goes first, then goes the second one, and so on up to n-th person.

Input

The first line contains two integers n and m ($1 \le n, m \le 1000$) — the number of passengers in the queue and the number of planes in the airport, correspondingly. The next line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le 1000$) — a_i stands for the number of empty seats in the i-th plane before the ticket office starts selling tickets.

The numbers in the lines are separated by a space. It is guaranteed that there are at least n empty seats in total.

Output

Print two integers — the maximum and the minimum number of zlotys that the airport administration can earn, correspondingly.

Examples

input
4 3 2 1 1
output
5 5

input
4 3 2 2 2
output
7 6

Note

In the first test sample the number of passengers is equal to the number of empty seats, so regardless of the way the planes are chosen, the administration will earn the same sum.

In the second sample the sum is maximized if the 1-st person in the queue buys a ticket to the 1-st plane, the 2-nd person — to the 2-nd plane, the 3-rd person — to the 3-rd plane, the 4-th person — to the 1-st plane. The sum is minimized if the 1-st person in the queue buys a ticket to the 1-st plane, the 2-nd person — to the 1-st plane, the 3-rd person — to the 2-nd plane, the 4-th person — to the 2-nd plane.

C. Ice Skating

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

Bajtek is learning to skate on ice. He's a beginner, so his only mode of transportation is pushing off from a snow drift to the north, east, south or west and sliding until he lands in another snow drift. He has noticed that in this way it's impossible to get from some snow drifts to some other by any sequence of moves. He now wants to heap up some additional snow drifts, so that he can get from any snow drift to any other one. He asked you to find the minimal number of snow drifts that need to be created.

We assume that Bajtek can only heap up snow drifts at integer coordinates.

Input

The first line of input contains a single integer n ($1 \le n \le 100$) — the number of snow drifts. Each of the following n lines contains two integers x_i and y_i ($1 \le x_i, y_i \le 1000$) — the coordinates of the i-th snow drift.

Note that the north direction coincides with the direction of Oy axis, so the east direction coincides with the direction of the Ox axis. All snow drift's locations are distinct.

Output

Output the minimal number of snow drifts that need to be created in order for Bajtek to be able to reach any snow drift from any other one.

Examples

input
2 2 1 1 2
output
1

input
2 2 1 4 1
output
0

D. Blackboard Fibonacci

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

Fibonacci numbers are the sequence of integers: $f_0 = 0$, $f_1 = 1$, $f_2 = 1$, $f_3 = 2$, $f_4 = 3$, $f_5 = 5$, ..., $f_n = f_{n-2} + f_{n-1}$. So every next number is the sum of the previous two.

Bajtek has developed a nice way to compute Fibonacci numbers on a blackboard. First, he writes a 0. Then, below it, he writes a 1. Then he performs the following two operations:

- operation "T": replace the top number with the sum of both numbers;
- operation "B": replace the bottom number with the sum of both numbers.

If he performs n operations, starting with "T" and then choosing operations alternately (so that the sequence of operations looks like "TBTBTBTB..."), the last number written will be equal to f_{n+1}.

Unfortunately, Bajtek sometimes makes mistakes and repeats an operation two or more times in a row. For example, if Bajtek wanted to compute f_7, then he would want to do $n = 6$ operations: "TBTBTB". If he instead performs the sequence of operations "TTTBBT", then he will have made 3 mistakes, and he will incorrectly compute that the seventh Fibonacci number is 10. The number of mistakes in the sequence of operations is the number of neighbouring equal operations («TT» or «BB»).

You are given the number n of operations that Bajtek has made in an attempt to compute f_{n+1} and the number r that is the result of his computations (that is last written number). Find the minimum possible number of mistakes that Bajtek must have made and any possible sequence of n operations resulting in r with that number of mistakes.

Assume that Bajtek always correctly starts with operation "T".

Input

The first line contains the integers n and r ($1 \le n, r \le 10^6$).

Output

The first line of the output should contain one number — the minimum possible number of mistakes made by Bajtek. The second line should contain n characters, starting with "T", describing one possible sequence of operations with that number of mistakes. Each character must be either "T" or "B".

If the required sequence doesn't exist, output "IMPOSSIBLE" (without quotes).

Examples

input
6 10
output
2 TBBTTB

input
4 5
output
0 TBTB

input
2 1
output
IMPOSSIBLE

E. Formurosa

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

The Bytelandian Institute for Biological Research (BIBR) is investigating the properties of two species of bacteria, named simply 0 and 1. Even under a microscope, bacteria of those two species are very difficult to distinguish. In fact, the only thing the scientists possess that is able to differentiate between them is a plant called Formurosa.

If the scientists place a sample of colonies of bacteria on each on Formurosa's leaves, it will activate a complicated nutrition process. During that process color of Formurosa changes to reflect the result of a — possibly very complicated — logical formula on the species of bacteria, involving constants and the operators | (OR), & (AND) and ^ (XOR). If it is 0, the plant will turn red, otherwise — it will turn blue.

For example, if the nutrition process of Formurosa is described by the formula: $(((?^?)|?)\&(1^?))$; then Formurosa has four leaves (the "?" signs denote the leaves). If we place $0, 1, 0, 0$ on the respective leaves, the result of the nutrition process will be $(((0^1)|0)\&(1^0)) = 1$, therefore the plant will turn blue.

The scientists have n colonies of bacteria. They do not know their types; the only thing they know for sure is that **not all colonies are of the same type**. They want to attempt to determine the bacteria's species by repeated evaluations with Formurosa. During each evaluation they must place exactly one sample on every leaf of the plant. However, they may use multiple samples of one colony during a single evaluation; they can even cover the whole plant with bacteria from one colony!

Is it possible for them to always determine the species of each colony, no matter what they are (assuming they are not all the same)?

Input

The first line of input contains a single integer n ($2 \le n \le 10^6$) — the number of colonies of bacteria.

The second line contains the formula describing the nutrition process of Formurosa. This line contains only characters «0», «1», «?», «|», «&», «^», «(», «)» and complies with the following grammar:

$$s \to 0|1|?|(s|s)|(s\&s)|(s\hat{}s)$$

The formula consists of no more than 10^6 characters.

Output

If it is always possible to determine the species of each colony, output "YES" (without quotes). Otherwise, output "NO" (without quotes).

Examples

input
2 (?^?)
output
NO

input
10 ?
output
YES

input
2 ((?^?)&?)
output
YES