Codeforces Round #134 (Div. 1)

A. 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.

Sample test(s)

input
2 2 1 1 2
output
1

input
2 2 1 4 1
output
0

B. 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: $f_0 = 0, f_1 = 1, f_2 = 1, f_3 = 2, f_4 = 3, f_5 = 5, \ldots, f_n = f_{n-2} + f_{n-1}$. 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).

Sample test(s)

input
6 10

output
2 TBBTTB

input
4 5

output
0 TBTB

input
2 1

output
IMPOSSIBLE

C. 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).

Sample test(s)

input
2 (?^?)
output
NO

input
10 ?
output
YES

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

D. Bitonix' Patrol

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

Byteland is trying to send a space mission onto the Bit-X planet. Their task is complicated by the fact that the orbit of the planet is regularly patrolled by Captain Bitonix, the leader of the space forces of Bit-X.

There are n stations around Bit-X numbered clockwise from 1 to n. The stations are evenly placed on a circular orbit, so the stations number i and $i+1$ $(1 \le i < n)$, and the stations number 1 and n, are neighboring. The distance between every pair of adjacent stations is equal to m space miles. To go on a patrol, Captain Bitonix jumps in his rocket at one of the stations and flies in a circle, covering a distance of at least one space mile, before finishing in some (perhaps the starting) station.

Bitonix' rocket moves by burning fuel tanks. After Bitonix attaches an x-liter fuel tank and chooses the direction (clockwise or counter-clockwise), the rocket flies exactly x space miles along a circular orbit in the chosen direction. Note that the rocket has no brakes; it is not possible for the rocket to stop before depleting a fuel tank.

For example, assume that $n = 3$ and $m = 60$ and Bitonix has fuel tanks with volumes of 10, 60, 90 and 100 liters. If Bitonix starts from station 1, uses the 100-liter fuel tank to go clockwise, then uses the 90-liter fuel tank to go clockwise, and then uses the 10-liter fuel tank to go counterclockwise, he will finish back at station 1. This constitutes a valid patrol. Note that Bitonix does not have to use all available fuel tanks. Another valid option for Bitonix in this example would be to simply use the 60-liter fuel tank to fly to either station 2 or 3.

However, if n was equal to 3, m was equal to 60 and the only fuel tanks available to Bitonix were one 10-liter tank and one 100-liter tank, he would have no way of completing a valid patrol (he wouldn't be able to finish any patrol exactly at the station).

The Byteland space agency wants to destroy some of Captain Bitonix' fuel tanks so that he cannot to complete any valid patrol. Find how many different subsets of the tanks the agency can destroy to prevent Captain Bitonix from completing a patrol and output the answer modulo 1000000007 $(10^9 + 7)$.

Input

The first line of the input contains three integers n $(2 \le n \le 1000)$ — the number of stations, m $(1 \le m \le 120)$ — the distance between adjacent stations, and t $(1 \le t \le 10000)$ — the number of fuel tanks owned by Captain Bitonix.

The second line of the input contains t space-separated integers between 1 and 10^9, inclusive — the volumes of Bitonix' fuel tanks.

Output

Output a single number — the number of distinct subsets of tanks that the Bytelandian space agency can destroy in order to prevent Captain Bitonix from completing a patrol, modulo $10^9 + 7$.

Sample test(s)

input
7 6 5 5 4 12 6 5
output
6

input
3 60 2 10 100
output
4

Note

All the fuel tanks are distinct, even if some of them have the same capacity.

E. Alien DNA

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

Professor Bajtocy is conducting experiments on alien DNA. He has discovered that it is subject to repetitive mutations — each mutation happens in the same way: some continuous subsequence of the alien DNA becomes active, copies itself, the copy gets mangled and inserts itself right after the original subsequence. The mangled copy of the activated continuous subsequence is formed by first joining all the elements at the even positions in that subsequence, and then joining all the elements at the odd ones at the end. That is, if the activated subsequence consists of 11 elements and represented as $s_1s_2\ldots s_{11}$, its mangled copy is $s_2s_4s_6s_8s_{10}s_1s_3s_5s_7s_9s_{11}$.

For example, if the original sequence was "ACTGG" and the mutation happened on the segment $[2, 4]$ (that is the activated subsequence is "CTG"), the mutated DNA is: "ACTG**TCG**G". The mangled copy of the activated subsequence is marked with bold font.

Professor Bajtocy has written down the original DNA sequence and the mutations that sequentially happened to it, and he now asks you to recover the first k elements of the DNA sequence after all the mutations.

Input

The first line of input contains the original DNA sequence, consisting only of letters "A", "C", "T" and "G" and not exceeding $3 \cdot 10^6$ in length.

The second line contains a single integer k ($1 \le k \le 3 \cdot 10^6$).

The third line contains a single integer n ($0 \le n \le 5000$) — the number of mutations. The next n lines describe the mutations in chronological order — each mutation is described by two numbers l_i and r_i ($1 \le l_i \le r_i \le 10^9$), meaning that the continuous subsequence $[l_i, r_i]$ has become active and cloned itself, joining itself with the mangled copy.

It is guaranteed that the input data is correct, that is, no mutation acts on non-existing elements of the DNA sequence, and the resulting DNA sequence has at least k elements.

Assume that the DNA elements are indexed starting from 1 and that the notation $[l, r]$ meaning the continuous subsequence of DNA sequence that consists of $r - l + 1$ elements starting at the l-th DNA sequence element and ending at the r-th DNA sequence element.

Output

Output a single line, containing the first k letters of the mutated DNA sequence.

Sample test(s)

input
GAGA 4 0

output
GAGA

input
ACGTACGT 16 2 1 2 2 8

output
ACCAGTACCGACATCG

Note

In the second example, after the first mutation the sequence is "ACCAGTACGT". After the second mutation it's "ACCAGTACCGACATCGT".