Educational Codeforces Round 51 (Rated for Div. 2)

A. Vasya And Password

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

Vasya came up with a password to register for *EatForces* — a string s. The password in *EatForces* should be a string, consisting of lowercase and uppercase Latin letters and digits.

But since *EatForces* takes care of the security of its users, user passwords must contain at least one digit, at least one uppercase Latin letter and at least one lowercase Latin letter. For example, the passwords "abaCABA12", "Z7q" and "3R24m" are valid, and the passwords "qwerty", "qwerty12345" and "Password" are not.

A substring of string s is a string $x = s_l s_{l+1} \dots s_{l+len-1} (1 \le l \le |s|, 0 \le len \le |s| - l + 1)$. len is the length of the substring. Note that the empty string is also considered a substring of s, it has the length 0.

Vasya's password, however, may come too weak for the security settings of *EatForces*. He likes his password, so he wants to replace some its substring with another string of the same length in order to satisfy the above conditions. This operation should be performed **exactly** once, and **the chosen string should have the minimal possible length**.

Note that the length of s should not change after the replacement of the substring, and the string itself should contain only lowercase and uppercase Latin letters and digits.

Input

The first line contains a single integer T $(1 \le T \le 100)$ — the number of testcases.

Each of the next T lines contains the initial password s $(3 \le |s| \le 100)$, consisting of lowercase and uppercase Latin letters and digits.

Only $T = 1$ is allowed for hacks.

Output

For each testcase print a renewed password, which corresponds to given conditions.

The length of the replaced substring is calculated as following: write down all the changed positions. If there are none, then the length is 0. Otherwise the length is the difference between the first and the last changed position plus one. For example, the length of the changed substring between the passwords "abcdef" $\to$ "a7cdEf" is 4, because the changed positions are 2 and 5, thus $(5 - 2) + 1 = 4$.

It is guaranteed that such a password always exists.

If there are several suitable passwords — output any of them.

Example

input

```
2
abcDCE
htQw27
```

output

```
abcD4E
htQw27
```

Note

In the first example Vasya's password lacks a digit, he replaces substring "C" with "4" and gets password "abcD4E". That means, he changed the substring of length 1.

In the second example Vasya's password is ok from the beginning, and nothing has to be changed. That is the same as replacing the empty substring with another empty substring (length 0).

B. Relatively Prime Pairs

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

You are given a set of all integers from l to r inclusive, $l < r$, $(r - l + 1) \le 3 \cdot 10^5$ and $(r - l)$ is always odd.

You want to split these numbers into exactly $\frac{r-l+1}{2}$ pairs in such a way that for each pair (i, j) the greatest common divisor of i and j is equal to 1. Each number should appear in exactly one of the pairs.

Print the resulting pairs or output that no solution exists. If there are multiple solutions, print any of them.

Input

The only line contains two integers l and r ($1 \le l < r \le 10^{18}$, $r - l + 1 \le 3 \cdot 10^5$, $(r - l)$ is odd).

Output

If any solution exists, print "YES" in the first line. Each of the next $\frac{r-l+1}{2}$ lines should contain some pair of integers. *GCD* of numbers in each pair should be equal to 1. All $(r - l + 1)$ numbers should be pairwise distinct and should have values from l to r inclusive.

If there are multiple solutions, print any of them.

If there exists no solution, print "NO".

Example

input

```
1 8
```

output

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

C. Vasya and Multisets

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

Vasya has a multiset s consisting of n integer numbers. Vasya calls some number x nice if it appears in the multiset exactly once. For example, multiset $\{1, 1, 2, 3, 3, 3, 4\}$ contains nice numbers 2 and 4.

Vasya wants to split multiset s into two multisets a and b **(one of which may be empty)** in such a way that the quantity of nice numbers in multiset a would be the same as the quantity of nice numbers in multiset b (the quantity of numbers to appear exactly once in multiset a and the quantity of numbers to appear exactly once in multiset b).

Input

The first line contains a single integer n $(2 \le n \le 100)$.

The second line contains n integers $s_1, s_2, \dots s_n$ $(1 \le s_i \le 100)$ — the multiset s.

Output

If there exists no split of s to satisfy the given requirements, then print "NO" in the first line.

Otherwise print "YES" in the first line.

The second line should contain a string, consisting of n characters. i-th character should be equal to 'A' if the i-th element of multiset s goes to multiset a and 'B' if if the i-th element of multiset s goes to multiset b. **Elements are numbered from 1 to n in the order they are given in the input.**

If there exist multiple solutions, then print any of them.

Examples

input

```
4
3 5 7 1
```

output

```
YES
BABA
```

input

```
3
3 5 1
```

output

```
NO
```

D. Bicolorings

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a grid, consisting of 2 rows and n columns. Each cell of this grid should be colored either black or white.

Two cells are considered neighbours if they have a **common border** and share the same color. Two cells A and B belong to the same component if they are neighbours, or if there is a neighbour of A that belongs to the same component with B.

Let's call some bicoloring *beautiful* if it has exactly k components.

Count the number of *beautiful* bicolorings. The number can be big enough, so print the answer modulo 998244353.

Input

The only line contains two integers n and k ($1 \le n \le 1000$, $1 \le k \le 2n$) — the number of columns in a grid and the number of components required.

Output

Print a single integer — the number of *beautiful* bicolorings modulo 998244353.

Examples

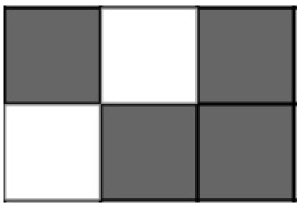
input
```
3 4
```
output
```
12
```

input
```
4 1
```
output
```
2
```

input
```
1 2
```
output
```
2
```

Note

One of possible bicolorings in sample 1:



E. Vasya and Big Integers

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Vasya owns three big integers — a, l, r. Let's define a partition of x such a sequence of strings $s_1, s_2, \dots, s_k$ that $s_1 + s_2 + \dots + s_k = x$, where $+$ is a concatanation of strings. s_i is the i-th element of the partition. For example, number 12345 has the following partitions: ["1", "2", "3", "4", "5"], ["123", "4", "5"], ["1", "2345"], ["12345"] and lots of others.

Let's call some partition of a *beautiful* if each of its elements **contains no leading zeros**.

Vasya want to know the number of *beautiful* partitions of number a, which has each of s_i satisfy the condition $l \le s_i \le r$. Note that the comparison is the integer comparison, not the string one.

Help Vasya to count the amount of partitions of number a such that they match all the given requirements. The result can be rather big, so print it modulo 998244353.

Input

The first line contains a single integer a $(1 \le a \le 10^{1000000})$.

The second line contains a single integer l $(0 \le l \le 10^{1000000})$.

The third line contains a single integer r $(0 \le r \le 10^{1000000})$.

It is guaranteed that $l \le r$.

It is also guaranteed that numbers a, l, r **contain no leading zeros**.

Output

Print a single integer — the amount of partitions of number a such that they match all the given requirements modulo 998244353.

Examples

input

```
135
1
15
```

output

```
2
```

input

```
10000
0
9
```

output

```
1
```

Note

In the first test case, there are two good partitions $13+5$ and $1+3+5$.

In the second test case, there is one good partition $1+0+0+0+0$.

F. The Shortest Statement

time limit per test: 4 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a weighed undirected **connected** graph, consisting of n vertices and m edges.

You should answer q queries, the i-th query is to find the shortest distance between vertices u_i and v_i.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5, m - n \le 20)$ — the number of vertices and edges in the graph.

Next m lines contain the edges: the i-th edge is a triple of integers v_i, u_i, d_i $(1 \le u_i, v_i \le n, 1 \le d_i \le 10^9, u_i \neq v_i)$. This triple means that there is an edge between vertices u_i and v_i of weight d_i. It is guaranteed that graph contains no self-loops and multiple edges.

The next line contains a single integer q $(1 \le q \le 10^5)$ — the number of queries.

Each of the next q lines contains two integers u_i and v_i $(1 \le u_i, v_i \le n)$ — descriptions of the queries.

Pay attention to the restriction $m - n \le 20$.

Output

Print q lines.

The i-th line should contain the answer to the i-th query — the shortest distance between vertices u_i and v_i.

Examples

input

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

output

```
3
4
1
```

input

```
8 13
1 2 4
2 3 6
```

```
3 4 1
4 5 12
5 6 3
6 7 8
7 8 7
1 4 1
1 8 3
2 6 9
2 7 1
4 6 3
6 8 2
8
1 5
1 7
2 3
2 8
3 7
3 4
6 8
7 8
```

output

```
7
5
6
7
7
1
2
7
```

G. Distinctification

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Suppose you are given a sequence S of k pairs of integers $(a_1, b_1), (a_2, b_2), \dots, (a_k, b_k)$.

You can perform the following operations on it:

1. Choose some position i and **increase** a_i by 1. That can be performed only if there exists at least one such position j that $i \neq j$ and $a_i = a_j$. The cost of this operation is b_i;
2. Choose some position i and **decrease** a_i by 1. That can be performed only if there exists at least one such position j that $a_i = a_j + 1$. The cost of this operation is $-b_i$.

Each operation can be performed arbitrary number of times (possibly zero).

Let $f(S)$ be minimum possible x such that there exists a sequence of operations with total cost x, after which all a_i from S are pairwise distinct.

Now for the task itself ...

You are given a sequence P consisting of n pairs of integers $(a_1, b_1), (a_2, b_2), \dots, (a_n, b_n)$. All b_i are pairwise distinct. Let P_i be the sequence consisting of the first i pairs of P. Your task is to calculate the values of $f(P_1), f(P_2), \dots, f(P_n)$.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of pairs in sequence P.

Next n lines contain the elements of P: i-th of the next n lines contains two integers a_i and b_i ($1 \le a_i \le 2 \cdot 10^5$, $1 \le b_i \le n$). It is guaranteed that all values of b_i are pairwise distinct.

Output

Print n integers — the i-th number should be equal to $f(P_i)$.

Examples

input

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

output

```
0
0
0
-5
-16
```

input
4 2 4 2 3 2 2 1 1

output
0 3 7 1