Codeforces Round #275 (Div. 2)

A. Counterexample

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

Your friend has recently learned about coprime numbers. A pair of numbers $\{a, b\}$ is called *coprime* if the maximum number that divides both a and b is equal to one.

Your friend often comes up with different statements. He has recently supposed that if the pair (a, b) is coprime and the pair (b, c) is coprime, then the pair (a, c) is coprime.

You want to find a counterexample for your friend's statement. Therefore, your task is to find three distinct numbers (a, b, c), for which the statement is false, and the numbers meet the condition $l \le a < b < c \le r$.

More specifically, you need to find three numbers (a, b, c), such that $l \le a < b < c \le r$, pairs (a, b) and (b, c) are coprime, and pair (a, c) is not coprime.

Input

The single line contains two positive space-separated integers l, r ($1 \le l \le r \le 10^{18}$; $r - l \le 50$).

Output

Print three positive space-separated integers a, b, c — three distinct numbers (a, b, c) that form the counterexample. If there are several solutions, you are allowed to print any of them. The numbers must be printed in ascending order.

If the counterexample does not exist, print the single number -1.

Sample test(s)

input
2 4
output
2 3 4

input
10 11
output
-1

input
900000000000000009 900000000000000029
output
900000000000000009 900000000000000010 900000000000000021

Note

In the first sample pair $(2, 4)$ is not coprime and pairs $(2, 3)$ and $(3, 4)$ are.

In the second sample you cannot form a group of three distinct integers, so the answer is -1.

In the third sample it is easy to see that numbers 900000000000000009 and 900000000000000021 are divisible by three.

B. Friends and Presents

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

You have two friends. You want to present each of them several positive integers. You want to present cnt_1 numbers to the first friend and cnt_2 numbers to the second friend. Moreover, you want all presented numbers to be distinct, that also means that no number should be presented to both friends.

In addition, the first friend does not like the numbers that are divisible without remainder by prime number x. The second one does not like the numbers that are divisible without remainder by prime number y. Of course, you're not going to present your friends numbers they don't like.

Your task is to find such minimum number v, that you can form presents using numbers from a set $1, 2, ..., v$. Of course you may choose not to present some numbers at all.

A positive integer number greater than 1 is called *prime* if it has no positive divisors other than 1 and itself.

Input

The only line contains four positive integers cnt_1, cnt_2, x, y ($1 \le cnt_1, cnt_2 < 10^9$; $cnt_1 + cnt_2 \le 10^9$; $2 \le x < y \le 3 \cdot 10^4$) — the numbers that are described in the statement. It is guaranteed that numbers x, y are prime.

Output

Print a single integer — the answer to the problem.

Sample test(s)

input
3 1 2 3
output
5

input
1 3 2 3
output
4

Note

In the first sample you give the set of numbers $\{1, 3, 5\}$ to the first friend and the set of numbers $\{2\}$ to the second friend. Note that if you give set $\{1, 3, 5\}$ to the first friend, then we cannot give any of the numbers $1, 3, 5$ to the second friend.

In the second sample you give the set of numbers $\{3\}$ to the first friend, and the set of numbers $\{1, 2, 4\}$ to the second friend. Thus, the answer to the problem is 4.

C. Diverse Permutation

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

Permutation p is an ordered set of integers $p_1, p_2, ..., p_n$, consisting of n distinct positive integers not larger than n. We'll denote as n the length of permutation $p_1, p_2, ..., p_n$.

Your task is to find such permutation p of length n, that the group of numbers $|p_1 - p_2|, |p_2 - p_3|, ..., |p_{n-1} - p_n|$ has exactly k distinct elements.

Input

The single line of the input contains two space-separated positive integers n, k ($1 \le k < n \le 10^5$).

Output

Print n integers forming the permutation. If there are multiple answers, print any of them.

Sample test(s)

input
3 2
output
1 3 2

input
3 1
output
1 2 3

input
5 2
output
1 3 2 4 5

Note

By $|x|$ we denote the absolute value of number x.

D. Interesting Array

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

We'll call an array of n non-negative integers $a[1], a[2], ..., a[n]$ *interesting*, if it meets m constraints. The i-th of the m constraints consists of three integers l_i, r_i, q_i ($1 \le l_i \le r_i \le n$) meaning that value $a[l_i] \ \& \ a[l_i + 1] \ \& \ \ldots \ \& \ a[r_i]$ should be equal to q_i.

Your task is to find any *interesting* array of n elements or state that such array doesn't exist.

Expression $x\&y$ means the bitwise AND of numbers x and y. In programming languages C++, Java and Python this operation is represented as "&", in Pascal — as "and".

Input

The first line contains two integers n, m ($1 \le n \le 10^5$, $1 \le m \le 10^5$) — the number of elements in the array and the number of limits.

Each of the next m lines contains three integers l_i, r_i, q_i ($1 \le l_i \le r_i \le n$, $0 \le q_i < 2^{30}$) describing the i-th limit.

Output

If the *interesting* array exists, in the first line print "YES" (without the quotes) and in the second line print n integers $a[1], a[2], ..., a[n]$ ($0 \le a[i] < 2^{30}$) decribing the *interesting* array. If there are multiple answers, print any of them.

If the *interesting* array doesn't exist, print "NO" (without the quotes) in the single line.

Sample test(s)

input
3 1 1 3 3
output
YES 3 3 3

input
3 2 1 3 3 1 3 2
output
NO

E. Game with Strings

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

You play the game with your friend. The description of this game is listed below.

Your friend creates n distinct strings of the same length m and tells you all the strings. Then he randomly chooses one of them. He chooses strings equiprobably, i.e. the probability of choosing each of the n strings equals $\frac{1}{n}$. You want to guess which string was chosen by your friend.

In order to guess what string your friend has chosen, you are allowed to ask him questions. Each question has the following form: «What character stands on position pos in the string you have chosen?» A string is considered guessed when the answers to the given questions uniquely identify the string. After the string is guessed, you stop asking questions.

You do not have a particular strategy, so as each question you equiprobably ask about a position that hasn't been yet mentioned. Your task is to determine the expected number of questions needed to guess the string chosen by your friend.

Input

The first line contains a single integer n ($1 \le n \le 50$) — the number of strings your friend came up with.

The next n lines contain the strings that your friend has created. It is guaranteed that all the strings are distinct and only consist of large and small English letters. Besides, the lengths of all strings are the same and are between 1 to 20 inclusive.

Output

Print the single number — the expected value. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-9}.

Sample test(s)

input
2 aab aac
output
2.000000000000000

input
3 aaA aBa Caa
output
1.666666666666667

input
3 aca vac wqq
output
1.000000000000000

Note

In the first sample the strings only differ in the character in the third position. So only the following situations are possible:

- you guess the string in one question. The event's probability is $\frac{1}{3}$;
- you guess the string in two questions. The event's probability is $\frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}$ (as in this case the first question should ask about the position that is other than the third one);
- you guess the string in three questions. The event's probability is $\frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{1} = \frac{1}{3}$;

Thus, the expected value is equal to $\frac{1+2+3}{3} = 2$

In the second sample we need at most two questions as any pair of questions uniquely identifies the string. So the expected number of questions is $\frac{1}{3} + 2 \cdot \frac{2}{3} = \frac{5}{3}$.

In the third sample whatever position we ask about in the first question, we immediately identify the string.