MemSQL start[c]up Round 2

A. Banana

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

Piegirl is buying stickers for a project. Stickers come on sheets, and each sheet of stickers contains exactly n stickers. Each sticker has exactly one character printed on it, so a sheet of stickers can be described by a string of length n. Piegirl wants to create a string s using stickers. She may buy as many sheets of stickers as she wants, and may specify any string of length n for the sheets, but all the sheets must be identical, so the string is the same for all sheets. Once she attains the sheets of stickers, she will take some of the stickers from the sheets and arrange (in any order) them to form s. Determine the minimum number of sheets she has to buy, and provide a string describing a possible sheet of stickers she should buy.

Input

The first line contains string s ($1 \le |s| \le 1000$), consisting of lowercase English characters only. The second line contains an integer n ($1 \le n \le 1000$).

Output

On the first line, print the minimum number of sheets Piegirl has to buy. On the second line, print a string consisting of n lower case English characters. This string should describe a sheet of stickers that Piegirl can buy in order to minimize the number of sheets. If Piegirl cannot possibly form the string s, print instead a single line with the number -1.

Examples

input
banana 4
output
2 baan

input
banana 3
output
3 nab

input
banana 2
output
-1

Note

In the second example, Piegirl can order 3 sheets of stickers with the characters "nab". She can take characters "nab" from the first sheet, "na" from the second, and "a" from the third, and arrange them to from "banana".

B. Palindrome

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

Given a string S, determine if it contains any palindrome of length exactly 100 as a **subsequence**. If it has any, print any one of them. If it doesn't have any, print a palindrome that is a subsequence of S and is as long as possible.

Input

The only line of the input contains one string S of length n ($1 \le n \le 5 \cdot 10^4$) containing only lowercase English letters.

Output

If S contains a palindrome of length exactly 100 as a subsequence, print any palindrome of length 100 which is a subsequence of S. If S doesn't contain any palindromes of length exactly 100, print a palindrome that is a subsequence of S and is as long as possible.

If there exists multiple answers, you are allowed to print any of them.

Examples

input
bbbabcbbb
output
bbbcbbb

input
rquwmzexectvnbanemsmdufrg
output
rumenanemur

Note

A subsequence of a string is a string that can be derived from it by deleting some characters without changing the order of the remaining characters. A palindrome is a string that reads the same forward or backward.

C. More Reclamation

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

In a far away land, there are two cities near a river. One day, the cities decide that they have too little space and would like to reclaim some of the river area into land.

The river area can be represented by a grid with r rows and exactly two columns — each cell represents a rectangular area. The rows are numbered 1 through r from top to bottom, while the columns are numbered 1 and 2.

Initially, all of the cells are occupied by the river. The plan is to turn some of those cells into land one by one, with the cities alternately choosing a cell to reclaim, and continuing until no more cells can be reclaimed.

However, the river is also used as a major trade route. The cities need to make sure that ships will still be able to sail from one end of the river to the other. More formally, if a cell (r, c) has been reclaimed, it is not allowed to reclaim any of the cells $(r - 1, 3 - c)$, $(r, 3 - c)$, or $(r + 1, 3 - c)$.

The cities are not on friendly terms, and each city wants to be the last city to reclaim a cell (they don't care about how many cells they reclaim, just who reclaims a cell last). The cities have already reclaimed n cells. Your job is to determine which city will be the last to reclaim a cell, assuming both choose cells optimally from current moment.

Input

The first line consists of two integers r and n ($1 \le r \le 100, 0 \le n \le r$). Then n lines follow, describing the cells that were already reclaimed. Each line consists of two integers: r_i and c_i ($1 \le r_i \le r, 1 \le c_i \le 2$), which represent the cell located at row r_i and column c_i. All of the lines describing the cells will be distinct, and the reclaimed cells will not violate the constraints above.

Output

Output "WIN" if the city whose turn it is to choose a cell can guarantee that they will be the last to choose a cell. Otherwise print "LOSE".

Examples

input
3 1 1 1
output
WIN

input
12 2 4 1 8 1
output
WIN

input
1 1 1 2
output
LOSE

Note

In the first example, there are 3 possible cells for the first city to reclaim: $(2, 1)$, $(3, 1)$, or $(3, 2)$. The first two possibilities both lose, as they leave exactly one cell for the other city.

However, reclaiming the cell at $(3, 2)$ leaves no more cells that can be reclaimed, and therefore the first city wins.

In the third example, there are no cells that can be reclaimed.

D. Rectangles and Square

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given n rectangles, labeled 1 through n. The corners of rectangles have integer coordinates and their edges are parallel to the Ox and Oy axes. The rectangles may touch each other, but they do not overlap (that is, there are no points that belong to the interior of more than one rectangle).

Your task is to determine if there's a non-empty subset of the rectangles that forms a square. That is, determine if there exists a subset of the rectangles and some square for which every point that belongs to the interior or the border of that square belongs to the interior or the border of at least one of the rectangles in the subset, and every point that belongs to the interior or the border of at least one rectangle in the subset belongs to the interior or the border of that square.

Input

First line contains a single integer n ($1 \le n \le 10^5$) — the number of rectangles. Each of the next n lines contains a description of a rectangle, with the i-th such line describing the rectangle labeled i. Each rectangle description consists of four integers: x_1, y_1, x_2, y_2 — coordinates of the bottom left and the top right corners ($0 \le x_1 < x_2 \le 3000$, $0 \le y_1 < y_2 \le 3000$).

No two rectangles overlap (that is, there are no points that belong to the interior of more than one rectangle).

Output

If such a subset exists, print "YES" (without quotes) on the first line of the output file, followed by k, the number of rectangles in the subset. On the second line print k numbers — the labels of rectangles in the subset in any order. If more than one such subset exists, print any one. If no such subset exists, print "NO" (without quotes).

Examples

input
9 0 0 1 9 1 0 9 1 1 8 9 9 8 1 9 8 2 2 3 6 3 2 7 3 2 6 7 7 5 3 7 6 3 3 5 6
output
YES 5 5 6 7 8 9

input
4 0 0 1 9 1 0 9 1 1 8 9 9 8 1 9 8
output
NO

Note

The first test case looks as follows:

Note that rectangles 6, 8, and 9 form a square as well, and would be an acceptable answer.

The second test case looks as follows:

E. Counting Skyscrapers

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

A number of skyscrapers have been built in a line. The number of skyscrapers was chosen uniformly at random between 2 and $314!$ (314 factorial, a very large number). The height of each skyscraper was chosen randomly and independently, with height i having probability 2^{-i} for all positive integers i. The floors of a skyscraper with height i are numbered 0 through $i - 1$.

To speed up transit times, a number of zip lines were installed between skyscrapers. Specifically, there is a zip line connecting the i-th floor of one skyscraper with the i-th floor of another skyscraper if and only if there are no skyscrapers between them that have an i-th floor.

Alice and Bob decide to count the number of skyscrapers.

Alice is thorough, and wants to know exactly how many skyscrapers there are. She begins at the leftmost skyscraper, with a counter at 1. She then moves to the right, one skyscraper at a time, adding 1 to her counter each time she moves. She continues until she reaches the rightmost skyscraper.

Bob is impatient, and wants to finish as fast as possible. He begins at the leftmost skyscraper, with a counter at 1. He moves from building to building using zip lines. At each stage Bob uses the highest available zip line to the right, but ignores floors with a height greater than h due to fear of heights. When Bob uses a zip line, he travels too fast to count how many skyscrapers he passed. Instead, he just adds 2^i to his counter, where i is the number of the floor he's currently on. He continues until he reaches the rightmost skyscraper.

Consider the following example. There are 6 buildings, with heights 1, 4, 3, 4, 1, 2 from left to right, and $h = 2$. Alice begins with her counter at 1 and then adds 1 five times for a result of 6. Bob begins with his counter at 1, then he adds 1, 4, 4, and 2, in order, for a result of 12. Note that Bob ignores the highest zip line because of his fear of heights ($h = 2$).

Bob's counter is at the top of the image, and Alice's counter at the bottom. All zip lines are shown. Bob's path is shown by the green dashed line and Alice's by the pink dashed line. The floors of the skyscrapers are numbered, and the zip lines Bob uses are marked with the amount he adds to his counter.

When Alice and Bob reach the right-most skyscraper, they compare counters. You will be given either the value of Alice's counter or the value of Bob's counter, and must compute the expected value of the other's counter.

Input

The first line of input will be a name, either string "`Alice`" or "`Bob`". The second line of input contains two integers n and h ($2 \le n \le 30000$, $0 \le h \le 30$). If the name is "`Alice`", then n represents the value of Alice's counter when she reaches the rightmost skyscraper, otherwise n represents the value of Bob's counter when he reaches the rightmost skyscraper; h represents the highest floor number Bob is willing to use.

Output

Output a single real value giving the expected value of the Alice's counter if you were given Bob's counter, or Bob's counter if you were given Alice's counter.

You answer will be considered correct if its absolute or relative error doesn't exceed 10^{-9}.

Examples

input
Alice 3 1
output
3.500000000

input
Bob 2 30
output
2

input
Alice 2572 10
output
3439.031415943

Note

In the first example, Bob's counter has a 62.5% chance of being 3, a 25% chance of being 4, and a 12.5% chance of being 5.

F. Buy One, Get One Free

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A nearby pie shop is having a special sale. For each pie you pay full price for, you may select one pie of a strictly lesser value to get for free. Given the prices of all the pies you wish to acquire, determine the minimum total amount you must pay for all of the pies.

Input

Input will begin with an integer n ($1 \leq n \leq 500000$), the number of pies you wish to acquire. Following this is a line with n integers, each indicating the cost of a pie. All costs are positive integers not exceeding 10^9.

Output

Print the minimum cost to acquire all the pies.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
6 3 4 5 3 4 5
output
14

input
5 5 5 5 5 5
output
25

input
4 309999 6000 2080 2080
output
314159

Note

In the first test case you can pay for a pie with cost 5 and get a pie with cost 4 for free, then pay for a pie with cost 5 and get a pie with cost 3 for free, then pay for a pie with cost 4 and get a pie with cost 3 for free.

In the second test case you have to pay full price for every pie.