Problem A. Colony of Bacteria

Time limit: 1 second
Memory limit: 512 megabytes

Scientists have discovered a new species of bacteria and have begun conducting experiments to study it.

In one of the experiments, they placed a colony of bacteria on an infinite grid, and it turned out that it expands every second. In every even second, the colony expands in eight directions, meaning it occupies cells adjacent to the occupied ones both orthogonally and diagonally, if they are not already occupied. In odd seconds, it only expands in four directions, occupying cells adjacent to the occupied ones orthogonally.

Help the scientists determine how many cells are occupied by the colony of bacteria at the k-th second of the experiment, given that it was placed on the grid in the first second.

Input

The first line contains an integer k — the time in seconds when the scientists want to know how many cells are occupied by the colony of bacteria ($1 \le k \le 10^8$).

Output

Output a single number – the number of cells occupied by the colony of bacteria at the k-th second of the experiment.

Examples

standard input	standard output
1	1
2	9
3	21
4	45
5	69

Note

The filling of the grid for the first five seconds, with the cell indicating the second when this cell will first be occupied by the colony of bacteria.

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

Problem B. Two-Story Advent Calendar

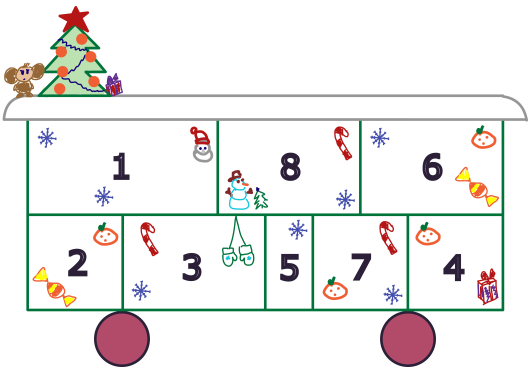
Time limit:	1 second
Memory limit:	512 megabytes

The "General Passenger Company" is launching New Year's train excursions from Saint Petersburg to Veliky Ustyug. For all purchasers of this trip, a special gift has been developed—an advent calendar.

The advent calendar is a box shaped like the main carrier of the "General Passenger Company" —a two-story train car. Inside the box, there are small boxes arranged in two levels, with each small box containing a candy. The upper level contains n small boxes, while the lower level contains m small boxes. Each small box is labeled with a natural number from 1 to $n+m$, inclusive. The numbers on the boxes do not repeat.

For each small box, its length is known. The boxes can differ in length. It is guaranteed that the sums of the lengths of the boxes on the first and second levels of the advent calendar are equal.

To properly open the advent calendar, on the first day, you must take and open the box numbered 1, on the second day —the box numbered 2, and so on, finishing the calendar with the box numbered $n+m$, which should be taken and opened on the $(n+m)$-th day. An example of the advent calendar is shown in the figure.



Advent calendar with 8 cells. To properly open it and create a New Year's mood, you need to open cell 1 8 days before New Year, cell 2 7 days before, and so on. On the last day —December 31 —you need to open cell 8.

Designer and perfectionist Maya decided to take a train ride for the New Year and received a two-story advent calendar as a gift. Maya finds it inconvenient when she opens a candy box on the lower level and there is at least one unopened box on top of it on the upper level.

Maya became curious about how many boxes she needs to remove from the calendar in advance to make it convenient. At the same time, Maya wants to leave as many boxes as possible in the advent calendar. Help her determine the minimum number of boxes that need to be removed from the calendar in advance so that when opening a box on the lower level, there are no unopened boxes from the upper level on top of it. Boxes can be removed in advance from both the upper and lower levels.

Input

The first line of input contains an integer n ($1 \le n \le 10^5$) —the number of boxes on the upper level of the calendar.

In the next n lines, there are two numbers a_i and x_i ($1 \le a_i \le 10^9$, $1 \le x_i \le n+m$) —the length of the i-th box on the upper level of the calendar and the number written on it, respectively.

On the $(n+1)$-th line of input, there is an integer m ($1 \le m \le 10^5$) —the number of boxes on the lower level of the calendar.

In the next m lines, there are two numbers b_j and y_j ($1 \le b_j \le 10^9$, $1 \le y_j \le n+m$) —the length of the j-th box on the lower level of the calendar and the number written on it, respectively.

It is guaranteed that $a_1 + a_2 + \ldots + a_n = b_1 + b_2 + \ldots + b_m$. It is guaranteed that all numbers in the set $\{x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m\}$ are distinct.

Output

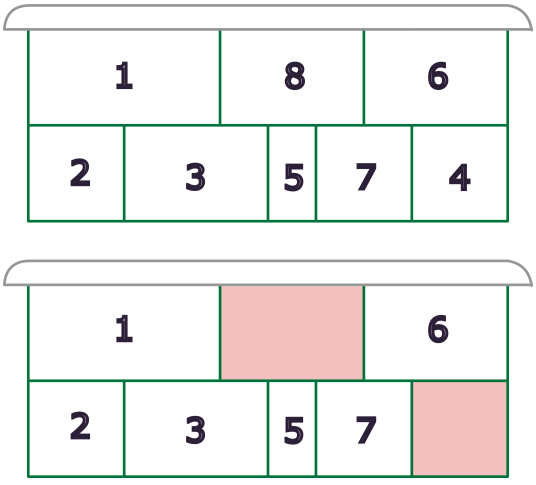
Output a single integer k—the minimum number of boxes that need to be removed from the calendar to make it convenient.

Examples

standard input	standard output
3 1 1 1 2 1 3 3 1 4 1 5 1 6	0
3 4 1 3 8 3 6 5 2 2 3 3 1 5 2 7 2 4	2

Note

In the second example, you can remove the boxes numbered 4 and 8. After this, the calendar will look like the one shown below and will become convenient.



Problem C. Intermediate Verticality

Time limit:	1 second
Memory limit:	512 megabytes

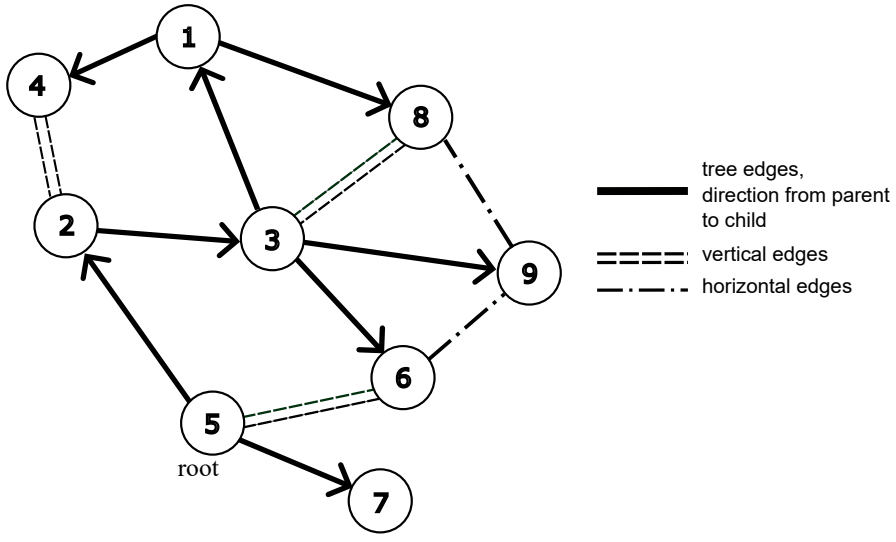
Two classical graph algorithms — depth-first search and breadth-first search — construct two spanning trees in a graph. The depth-first search is known for producing a tree that has no *horizontal* edges, which are edges connecting vertices that are not ancestors of each other, while breadth-first search is known for producing a tree that has no *vertical* edges — edges connecting a vertex to its ancestor in the tree. In this problem, you will need to construct an intermediate spanning tree that has a specified number of horizontal and vertical edges.

Recall that an undirected graph consists of a set of vertices V and a set of edges E, where each edge connects two vertices. We will consider connected graphs, where it is possible to reach any vertex from any other vertex via edges. A tree is a connected undirected graph that contains no cycles, and a spanning tree in a graph is a subset of its edges that forms a tree, allowing one to reach any vertex of the graph from any other. Recall two fundamental properties of a tree: in a tree with n vertices, there are exactly $n - 1$ edges, and there is exactly one path between any two vertices in a tree.

We will designate a vertex r in the graph, which we will call the *root* of the tree. The vertices that lie on the unique path from vertex x to vertex r are called the *ancestors* of vertex x, and the first vertex on this path is called the *parent* of vertex x and is denoted as p_x. The root has no parent.

If a root and a spanning tree are fixed in the graph, then all edges of the graph can be divided into three types:

- *tree edges* — the edges of the chosen spanning tree;
- *vertical edges* — the edges not belonging to the tree that connect a vertex to its ancestor;
- *horizontal edges* — the remaining edges of the graph.



The *verticality* of the spanning tree in the graph is defined as the number of vertical edges.

You are given a graph with n vertices and m edges, the root of the tree r, and a number h, $0 \le h \le m-n+1$. You need to construct a spanning tree of the given graph with root at r, having a verticality equal to h, or report that such a tree does not exist.

Input

Each test consists of several sets of input data. The first line contains one integer t — the number of sets of input data ($1 \le t \le 10^5$). Following this are descriptions of t sets of input data.

In the first line of each set of input data, there are four integers n, m, r, and h — the number of vertices and edges in the graph, respectively, the index of the root vertex, and the required verticality of the future spanning tree ($2 \le n \le 3 \cdot 10^5$; $n - 1 \le m \le 3 \cdot 10^5$; $1 \le r \le n$; $0 \le h \le m - n + 1$).

In each of the following m lines, there are two integers u_i, v_i — the indices of the vertices connected by an edge in the graph ($1 \le u_i, v_i \le n$; $u_i \ne v_i$).

It is guaranteed that all graphs are connected, contain no loops or multiple edges. It is guaranteed that the sum of n across all input data sets does not exceed $3 \cdot 10^5$. It is guaranteed that the sum of m across all input data sets does not exceed $3 \cdot 10^5$.

Output

For each test case, find the required spanning tree T and output in a separate line n integers $p_1, p_2, \ldots, p_n$, where p_i is the index of the parent of the i-th vertex in the tree T ($1 \le p_i \le n$). For p_r, you can output any number from 1 to n. If a tree T with the desired properties does not exist, output n numbers -1 instead.

Example

standard input	standard output
4 5 7 2 0 1 2 2 3 3 4 4 1 3 5 5 4 1 5 5 7 2 1 1 2 2 3 3 4 4 1 3 5 5 4 1 5 5 7 2 2 1 2 2 3 3 4 4 1 3 5 5 4 1 5 5 7 2 3 1 2 2 3 3 4 4 1 3 5 5 4 1 5	2 1 2 3 3 2 1 4 1 1 2 1 5 5 1 2 1 4 1 3

Problem D. Two Arrays

Time limit: 2 seconds
Memory limit: 512 megabytes

Let the maximum in the array d be denoted as $\max(d)$ and the minimum as $\min(d)$.

Two arrays a and b of length n are given. In one operation, you can choose an index $1 \le i \le n$ and simultaneously increase the elements a_i and b_i by one: $a_i = a_i + 1$, $b_i = b_i + 1$. It is necessary to use these operations to achieve the simultaneous fulfillment of two conditions:

- $\max(a) - \min(a) \le x$,
- $\max(b) - \min(b) \le y$.

Determine the minimum number of operations required to achieve the simultaneous fulfillment of the specified conditions, or find out that it is impossible.

Input

Each test consists of several test cases. The first line contains one integer t — the number of test cases ($1 \le t \le 10^5$). The description of the test cases follows.

The first line of each test case contains three integers: n, x, y ($1 \le n \le 10^5$, $0 \le x, y \le 10^9$).

The second line of each test case contains n integers $a_1, a_2, \ldots a_n$ — the elements of array a ($-10^9 \le a_i \le 10^9$).

The third line of each test case contains n integers $b_1, b_2, \ldots b_n$ — the elements of array b ($-10^9 \le b_i \le 10^9$).

It is guaranteed that the sum of n across all test cases does not exceed 10^5.

Output

For each test case, output one integer — the minimum possible number of operations required to satisfy both conditions. If it is impossible to satisfy both conditions simultaneously, output -1.

Example

standard input	standard output
5 4 2 3 -1 -2 -3 -4 -1 -2 -3 -4 3 3 2 1 6 4 1 4 1 4 0 3 0 2 1 2 0 2 3 3 5 2 1 -1 0 1 2 3 2 2 2 2 2 3 66 77 235 -111 9 100 -200 -100	1 3 3 -1 440

Problem E. Classics

Time limit: 1 second
Memory limit: 512 megabytes

You are probably familiar with the classic problem of finding the longest increasing subsequence in an array. Let a be an array consisting of n integers. A subsequence $i_1 < i_2 < \ldots < i_k$ is called *increasing* if $a_{i_1} < a_{i_2} < \ldots < a_{i_k}$. The longest increasing subsequence is the increasing subsequence of maximum length. Of course, we will not ask you to solve the classic problem; you will have to solve its more complicated version...

Initially, there is an empty array a. Then, the numbers $1, 2, \ldots, n$ are added to the array in this order. The number i is added to the array at position p_i. Positions in the array are numbered with integers from 1 to k, where k is the current size of the array. When adding an element at position p in an array of size k, all elements that previously had positions from p to k are shifted one position to the right, and the current element is added to the freed space.

Your task is to determine the length of the longest increasing subsequence in the array after each addition of a new element.

Input

The first line contains one integer n ($1 \le n \le 200\,000$) — the number of added elements.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le i$) — p_i denotes the position where element i is added.

Output

Output n integers — the length of the longest increasing subsequence of the array after each addition of a new element.

Examples

standard input	standard output
5 1 2 1 3 4	1 2 2 2 3
1 1	1

Note

The array in the first example changed as follows: $[] \to [1] \to [1, 2] \to [3, 1, 2] \to [3, 1, 4, 2] \to [3, 1, 4, 5, 2]$.

Problem F. Exchange and Deletion

Time limit: 1 second
Memory limit: 512 megabytes

There is a classic way to remove an element from an array: swap the values of the element to be removed and the last element of the array, and then delete the last element. Unfortunately, it turned out that this method does not always preserve the order of the elements in the array. Your task is to count the number of sequences of k deletions after which the initially sorted array in ascending order will remain sorted.

You are given an array a of length n, initially filled with numbers from 1 to n in ascending order, that is, $a_i = i$, and an array b of length k, whose elements are pairwise distinct numbers from 1 to n.

There are k steps performed, at the j-th step the following occurs: an index i is chosen from 1 to $n-j+1$ such that $a_i = b_j$, after which a_i and a_{n-j+1} are swapped (if $i = n-j+1$, nothing happens). Then the last element of the array at this step, which has the index $n-j+1$, is removed from the array.

The array $[b_1, b_2, \ldots, b_k]$ is called *good* if after performing k steps the array $[a_1, a_2, \ldots, a_{n-k}]$ is strictly increasing.

You are given the numbers n and k, count the number of good arrays $[b_1, b_2, \ldots, b_k]$. The answer may be too large, so output the remainder of the answer when divided by $10^9 + 7$.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases. In the following t lines, the test cases are provided.

In the first line of each test case, the integers n and k are given ($1 \le n \le 5 \cdot 10^5$, $0 \le k \le n$).

It is guaranteed that the sum of n across all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output a single integer — the number of good arrays b, taken modulo $10^9 + 7$.

Example

standard input	standard output
5 1 0 1 1 2 2 3 1 4 2	1 1 2 2 7

Note

Let's analyze the fourth test from the example. In it, $n = 3$ and $k = 1$. Initially, $a = [1, 2, 3]$. Let's see how a changes after the first step for all possible arrays b.

- $b = [1]$. Then a changes as follows: $[1, 2, 3] \to [3, 2, 1] \to [3, 2]$. The array $[3, 2]$ is not increasing.
- $b = [2]$. Then a changes as follows: $[1, 2, 3] \to [1, 3, 2] \to [1, 3]$. The array $[1, 3]$ is increasing.
- $b = [3]$. Then a changes as follows: $[1, 2, 3] \to [1, 2, 3] \to [1, 2]$. The array $[1, 2]$ is increasing.

We find that there are two good arrays $b = [2]$ and $b = [3]$, so the answer to the fourth test from the example is 2.

Problem G. M-11 Highway

Time limit: 1 second
Memory limit: 512 megabytes

The new high-speed highway M-11 is an infinite straight line.

On the highway, there are n stopping points, each of which is a rest area or a gas station. Each stopping point is defined by its coordinate x_i, and no two stopping points are located at the same place. A triplet of stopping points (i, j, k) is called *convenient* if $x_i < x_j < x_k$, there are gas stations at points x_i and x_k, a rest area at point x_j, and the distance between the gas stations does not exceed d.

A team from Moscow is planning to travel to the contest along the M-11 highway, and its leader became curious about how many convenient triplets of stopping points exist along the way.

Input

The first line contains two natural numbers n and d — the number of stopping points and the maximum distance between gas stations ($3 \le n \le 5 \cdot 10^5$, $2 \le d \le 10^9$).

In the following n lines, the stopping points are given. Each stopping point is defined by two integers x_i and t_i — the coordinate of the point and its type. Type 0 denotes a rest area, and type 1 denotes a gas station ($-10^{18} \le x_i \le 10^{18}$; $t_i \in \{0, 1\}$). It is guaranteed that the coordinates of the stopping points are in increasing order.

Output

Output a single number — the number of convenient triplets.

Examples

standard input	standard output
8 5 1 1 2 0 3 1 6 0 7 0 8 1 15 1 19 1	3
10 6 0 1 1 0 3 1 4 0 5 1 8 1 10 0 11 0 14 1 18 1	7

Note

In the first input set, the convenient triplets are $(1, 2, 3)$, $(3, 4, 6)$, and $(3, 5, 6)$.

Problem H. Exploration Robots

Time limit: 1 second
Memory limit: 512 megabytes

The field for testing the robots consists of n cells numbered by integers 1 to n from left to right. Each cell contains a letter of the English alphabet; reading the letters from the first to the n-th cell forms the string s.

There are two robots that can move across the field, with the following conditions:

- The robots know the string s;
- The robots can freely exchange information;
- The robots always know the distance between them, as well as which robot is to the left and which is to the right;
- Each of the two robots can read the letter that is directly below it;

In one step, a robot can, after exchanging information with the other robot, move to an adjacent cell to the left or to an adjacent cell to the right. If a robot attempts to move left of the first cell or right of the n-th cell, it is destroyed.

The scientists want to conduct q experiments, in the i-th of which the first robot is placed at position x_i, and the second at position y_i. The goal of the robots in each experiment is to visit as many different cells as possible. For each experiment, it is necessary to determine the maximum number of cells that the robots can visit without risking destruction.

Input

The first line contains a pair of numbers n and q ($1 \le n, q \le 300\,000$) — the number of cells and the number of experiments.

The second line contains the string s of length n, consisting of n lowercase Latin letters.

In the following q lines, pairs of numbers x_i, y_i are given ($1 \le x_i, y_i \le n$).

Output

For each experiment, output the maximum number of different cells that the robots can visit.

Example

standard input	standard output
10 4 aabaabbaab 4 5 8 5 2 3 1 1	3 10 3 3

Note

Consider the last experiment in the example. The robots start at the same point and see the letter "a". They understand that moving left is dangerous, as it may lead to the destruction of the robot. However, moving right is safe, as the last letter of the string is "b". One or both robots move to the right until they reach the letter "b". Upon reaching the letter "b", the robots know that the string before it was "aab", which could be either the beginning or the end of the string; they cannot go beyond its limits without risking destruction, so they managed to visit a total of 3 cells.

Problem I. Prank

Time limit: 1 second
Memory limit: 512 megabytes

Katya formed the word s_1 from blocks, but when she returned to the room, she saw her brother Andrey running out. Now the word made from the blocks looked different — s_2. Andrey admitted that he played a little *prank*. His prank consisted of the following: Andrey would choose a position and then insert two blocks with the same letter next to it. He could place these two blocks at the beginning of the string, at the end of the string, or between two neighboring blocks.

Help Katya determine whether Andrey told the truth, that is, whether the string s_2 could have been obtained from the string s_1 by possibly applying several *pranks*.

Input

One test contains several sets of input data.

The first line contains one integer t — the number of sets of input data ($1 \le t \le 500\,000$).

In the first line of each set description, there is one string s_1 — the word from the blocks that Katya originally had.

In the second line of each set description, there is one string s_2 — the word from the blocks that Andrey obtained.

It is guaranteed that all words consist of lowercase Latin letters. The total length of all words does not exceed $1\,000\,000$.

Output

For each set of input data, output "`YES`", if Andrey could be telling the truth, and "`NO`" otherwise.

Example

standard input	standard output
2 hello havvaeeello test tesssst	YES NO

Problem J. Nightmare Sum

Time limit: 3 seconds
Memory limit: 512 megabytes

Given an array a of length n, consisting of distinct positive integers. Compute

$$\sum_{l=1}^{n}\sum_{r=l}^{n}\left\lfloor\frac{\max(a_l, a_{l+1}, \ldots, a_r)}{\min(a_l, a_{l+1}, \ldots, a_r)}\right\rfloor$$

Here, $\lfloor x \rfloor$ denotes x rounded down to the nearest integer.

Thus, it is necessary to compute the sum of the results of integer division of the maximum by the minimum over all subarrays of the array a.

Input

The first line of input contains a single integer n — the length of the array ($1 \le n \le 300\,000$).

The second line of input contains n integers — the array a ($1 \le a_i \le 300\,000$).

It is guaranteed that all numbers in the array a are distinct.

Output

Output a single number — the desired sum.

Example

standard input	standard output
6 1 3 6 4 2 5	56

Note

Let's consider the example in more detail:

$[l, r]$	$a[l \dots r]$	min	max	$\left\lfloor \frac{\max}{\min} \right\rfloor$
$[1, 1]$	$[1]$	1	1	1
$[1, 2]$	$[1, 3]$	1	3	3
$[1, 3]$	$[1, 3, 6]$	1	6	6
$[1, 4]$	$[1, 3, 6, 4]$	1	6	6
$[1, 5]$	$[1, 3, 6, 4, 2]$	1	6	6
$[1, 6]$	$[1, 3, 6, 4, 2, 5]$	1	6	6
$[2, 2]$	$[3]$	3	3	1
$[2, 3]$	$[3, 6]$	3	6	2
$[2, 4]$	$[3, 6, 4]$	3	6	2
$[2, 5]$	$[3, 6, 4, 2]$	2	6	3
$[2, 6]$	$[3, 6, 4, 2, 5]$	2	6	3
$[3, 3]$	$[6]$	6	6	1
$[3, 4]$	$[6, 4]$	4	6	1
$[3, 5]$	$[6, 4, 2]$	2	6	3
$[3, 6]$	$[6, 4, 2, 5]$	2	6	3
$[4, 4]$	$[4]$	4	4	1
$[4, 5]$	$[4, 2]$	2	4	2
$[4, 6]$	$[4, 2, 5]$	2	5	2
$[5, 5]$	$[2]$	2	2	1
$[5, 6]$	$[2, 5]$	2	5	2
$[6, 6]$	$[5]$	5	5	1

Problem K. Petya's Cryptography

Time limit: 1 second
Memory limit: 512 megabytes

Petya received a failing grade in cryptography, but it was the teachers who did not understand his genius, not him being lazy. To convince the entire world community of his genius, Petya created a new public key encryption system — *PSA*. Any self-respecting public key cryptosystem has a public key and a private key.

As a private key, Petya chose a tree T, and as a public key — two numbers (n, p), where n is the number of vertices in the tree T, and p is the number of paths of length 2 in T. Recall that a tree is an undirected connected graph that does not contain cycles.

The peculiarity of Petya's cryptosystem is that any private key corresponding to the public key will suffice to break it. But that's not a problem; Petya chose quite a complex task, right? Restore any private key of the cryptosystem *PSA* or state that such a public key could not have been produced.

Input

The input consists of a single line containing two numbers n and p ($1 \le n \le 1000$, $0 \le p \le 10^9$).

Output

If a solution exists, output "`Yes`" in the first line. In the next $n-1$ lines, output two distinct integers from 1 to n — the edges of the tree.

If no solution exists, output "`No`" in a single line.

Examples

standard input	standard output
7 11	Yes 1 2 2 3 3 4 3 5 3 6 3 7
5 5	No

Problem L. Two Scooters

Time limit: 1 second
Memory limit: 512 megabytes

Katya knows that the travel time from home to the metro on a scooter is t seconds.

The cost of the ride on the scooter from company `W` is calculated as follows: first, the number of full minutes spent on the trip is determined, then this time is multiplied by 60 and used to calculate the cost at the rate of c_1 cents per second.

The cost of the ride on the scooter from company `Y` is calculated differently: first, the cost of the trip is calculated at the rate of c_2 cents per second, and then this amount is rounded up to the nearest whole euro.

Help Katya understand what is the minimum cost she can get to the metro on a scooter from one of these companies.

Recall that there are 100 cents in one euro.

Input

The first line contains three integers t, c_1, and c_2 — the travel time in seconds, the fare in cents per second on the scooter from company `W`, and the fare in cents per second on the scooter from company `Y` ($1 \le t \le 1000$, $10 \le c_1, c_2 \le 20$).

Output

Output the minimum cost of the ride in cents.

Example

standard input	standard output
473 10 11	4200