Codeforces Round #530 (Div. 2)

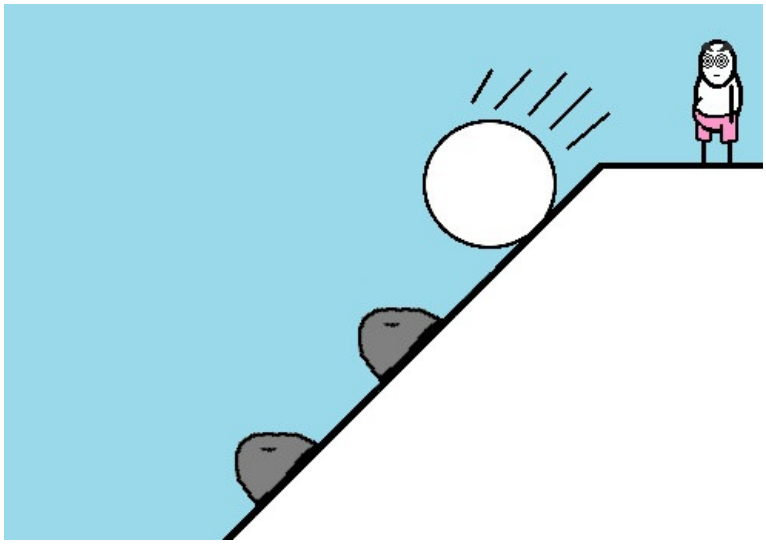
A. Snowball

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

Today's morning was exceptionally snowy. Meshanya decided to go outside and noticed a huge snowball rolling down the mountain! Luckily, there are two stones on that mountain.

Initially, snowball is at height h and it has weight w. Each second the following sequence of events happens: snowball's weights increases by i, where i — is the current height of snowball, then snowball hits the stone (if it's present at the current height), then snowball moves one meter down. If the snowball reaches height zero, it stops.

There are exactly two stones on the mountain. First stone has weight u_1 and is located at height d_1, the second one — u_2 and d_2 respectively. When the snowball hits either of two stones, it loses weight equal to the weight of that stone. If after this snowball has negative weight, then its weight becomes zero, but the snowball continues moving as before.



Find the weight of the snowball when it stops moving, that is, it reaches height 0.

Input

First line contains two integers w and h — initial weight and height of the snowball ($0 \le w \le 100$; $1 \le h \le 100$).

Second line contains two integers u_1 and d_1 — weight and height of the first stone ($0 \le u_1 \le 100$; $1 \le d_1 \le h$).

Third line contains two integers u_2 and d_2 — weight and heigth of the second stone ($0 \le u_2 \le 100$; $1 \le d_2 \le h$; $d_1 \ne d_2$). Notice that stones always have different heights.

Output

Output a single integer — final weight of the snowball after it reaches height 0.

Examples

input
4 3 1 1 1 2
output
8

input
4 3 9 2 0 1
output
1

Note

In the first example, initially a snowball of weight 4 is located at a height of 3, there are two stones of weight 1, at a height of 1 and

2, respectively. The following events occur sequentially:

- The weight of the snowball increases by 3 (current height), becomes equal to 7.
- The snowball moves one meter down, the current height becomes equal to 2.
- The weight of the snowball increases by 2 (current height), becomes equal to 9.
- The snowball hits the stone, its weight decreases by 1 (the weight of the stone), becomes equal to 8.
- The snowball moves one meter down, the current height becomes equal to 1.
- The weight of the snowball increases by 1 (current height), becomes equal to 9.
- The snowball hits the stone, its weight decreases by 1 (the weight of the stone), becomes equal to 8.
- The snowball moves one meter down, the current height becomes equal to 0.

Thus, at the end the weight of the snowball is equal to 8.

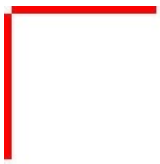
B. Squares and Segments

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

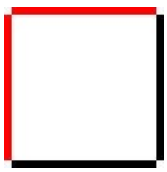
Little Sofia is in fourth grade. Today in the geometry lesson she learned about segments and squares. On the way home, she decided to draw n squares in the snow with a side length of 1. For simplicity, we assume that Sofia lives on a plane and can draw only segments of length 1, parallel to the coordinate axes, with vertices at integer points.

In order to draw a segment, Sofia proceeds as follows. If she wants to draw a vertical segment with the coordinates of the ends (x, y) and $(x, y+1)$. Then Sofia looks if there is already a drawn segment with the coordinates of the ends (x', y) and $(x', y+1)$ for some x'. If such a segment exists, then Sofia quickly draws a new segment, using the old one as a guideline. If there is no such segment, then Sofia has to take a ruler and measure a new segment for a long time. Same thing happens when Sofia wants to draw a horizontal segment, but only now she checks for the existence of a segment with the same coordinates x, $x+1$ and the differing coordinate y.

For example, if Sofia needs to draw one square, she will have to draw two segments using a ruler:

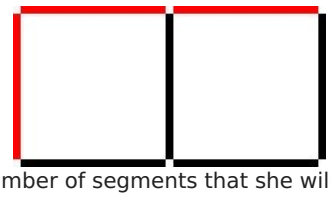


After that, she can draw the remaining two segments, using the first two as a guide:



If Sofia needs to draw two squares, she will have to draw three segments using a ruler:

After that, she can draw the remaining four segments, using the first three as a guide:



Sofia is in a hurry, so she wants to minimize the number of segments that she will have to draw with a ruler without a guide. Help her find this minimum number.

Input

The only line of input contains a single integer n ($1 \le n \le 10^9$), the number of squares that Sofia wants to draw.

Output

Print single integer, the minimum number of segments that Sofia will have to draw with a ruler without a guide in order to draw n squares in the manner described above.

Examples

input
1
output

2

input
2
output
3

input
4
output
4

C. Postcard

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

Andrey received a postcard from Irina. It contained only the words "Hello, Andrey!", and a strange string consisting of lowercase Latin letters, snowflakes and candy canes. Andrey thought that this string is an encrypted message, and decided to decrypt it.

Andrey noticed that snowflakes and candy canes always stand after the letters, so he supposed that the message was encrypted as follows. Candy cane means that the letter before it can be removed, or can be left. A snowflake means that the letter before it can be removed, left, or repeated several times.

For example, consider the following string:

This string can encode the message «happynewyear». For this, candy canes and snowflakes should be used as follows:

- candy cane 1: remove the letter w,
- snowflake 1: repeat the letter p twice,
- candy cane 2: leave the letter n,
- snowflake 2: remove the letter w,
- snowflake 3: leave the letter e.

Please note that the same string can encode different messages. For example, the string above can encode «hayewyar», «happpppynewwwwwyear», and other messages.

Andrey knows that messages from Irina usually have a length of k letters. Help him to find out if a given string can encode a message of k letters, and if so, give an example of such a message.

Input

The first line contains the string received in the postcard. The string consists only of lowercase Latin letters, as well as the characters «*» and «?», meaning snowflake and candy cone, respectively. These characters can only appear immediately after the letter. The length of the string does not exceed 200.

The second line contains an integer number k ($1 \le k \le 200$), the required message length.

Output

Print any message of length k that the given string can encode, or «Impossible» if such a message does not exist.

Examples

input
hw?ap*yn?eww*ye*ar 12
output
happynewyear

input
ab?a 2
output
aa

input
ab?a 3
output
aba

input
ababb 5
output
ababb

input
ab?a 1
output
Impossible

D. Sum in the tree

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

Mitya has a rooted tree with n vertices indexed from 1 to n, where the root has index 1. Each vertex v initially had an integer number $a_v \geq 0$ written on it. For every vertex v Mitya has computed s_v: the sum of all values written on the vertices on the path from vertex v to the root, as well as h_v — the depth of vertex v, which denotes the number of vertices on the path from vertex v to the root. Clearly, $s_1 = a_1$ and $h_1 = 1$.

Then Mitya erased all numbers a_v, and by accident he also erased all values s_v for vertices with even depth (vertices with even h_v). Your task is to restore the values a_v for every vertex, or determine that Mitya made a mistake. In case there are multiple ways to restore the values, you're required to find one which minimizes the total sum of values a_v for all vertices in the tree.

Input

The first line contains one integer n — the number of vertices in the tree ($2 \leq n \leq 10^5$). The following line contains integers p_2, p_3, ... p_n, where p_i stands for the parent of vertex with index i in the tree ($1 \leq p_i < i$). The last line contains integer values s_1, s_2, ..., s_n ($-1 \leq s_v \leq 10^9$), where erased values are replaced by -1.

Output

Output one integer — the minimum total sum of all values a_v in the original tree, or -1 if such tree does not exist.

Examples

input
5 1 1 1 1 1 -1 -1 -1 -1
output
1

input
5 1 2 3 1 1 -1 2 -1 -1
output
2

input
3 1 2 2 -1 1
output
-1

E. Nice table

time limit per test: 1 second
memory limit per test: 256 megabytes

input: standard input
output: standard output

You are given an $n \times m$ table, consisting of characters «A», «G», «C», «T». Let's call a table *nice*, if every 2×2 square contains all four distinct characters. Your task is to find a nice table (also consisting of «A», «G», «C», «T»), that differs from the given table in the minimum number of characters.

Input
First line contains two positive integers n and m — number of rows and columns in the table you are given ($2 \leq n, m, n \times m \leq 300\,000$). Then, n lines describing the table follow. Each line contains exactly m characters «A», «G», «C», «T».

Output
Output n lines, m characters each. This table must be nice and differ from the input table in the minimum number of characters.

Examples

input
2 2 AG CT
output
AG CT

input
3 5 AGCAG AGCAG AGCAG
output
TGCAT CATGC TGCAT

Note
In the first sample, the table is already nice. In the second sample, you can change 9 elements to make the table nice.

F. Cookies

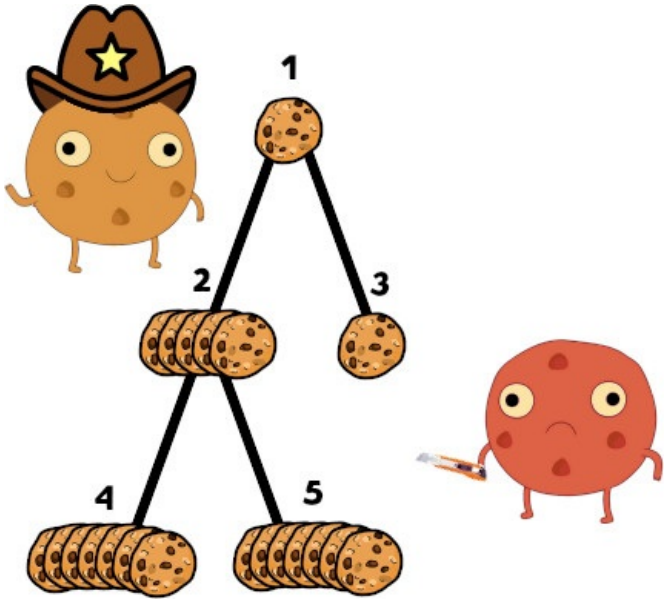
time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Mitya and Vasya are playing an interesting game. They have a rooted tree with n vertices, and the vertices are indexed from 1 to n. The root has index 1. Every other vertex $i \geq 2$ has its *parent* p_i, and vertex i is called a *child* of vertex p_i.

There are some cookies in every vertex of the tree: there are x_i cookies in vertex i. It takes exactly t_i time for Mitya to eat **one** cookie in vertex i. There is also a chip, which is initially located in the root of the tree, and it takes l_i time to move the chip along the edge connecting vertex i with its parent.

Mitya and Vasya take turns playing, Mitya goes first.

- Mitya moves the chip from the vertex, where the chip is located, to one of its children.
- Vasya can remove an edge from the vertex, where the chip is located, to one of its children. Vasya can also decide to skip his turn.



Mitya can stop the game at any his turn. Once he stops the game, he moves the chip up to the root, eating some cookies along his way. Mitya can decide how many cookies he would like to eat in every vertex on his way. The total time spent on descend, ascend and eating cookies should not exceed T. Please note that in the end of the game the chip is always located in the root of the tree: Mitya can not leave the chip in any other vertex, even if he has already eaten enough cookies — he must move the chip back to the root (and every move from vertex v to its parent takes l_v time).

Find out what is the maximum number of cookies Mitya can eat, regardless of Vasya's actions.

Input

The first line contains two integers n and T — the number of vertices in the tree and the time he has to accomplish his task ($2 \leq n \leq 10^5$; $1 \leq T \leq 10^{18}$).

The second line contains n integers x_1, x_2, ..., x_n — number of cookies located in the corresponding vertex ($1 \leq x_i \leq 10^6$). The third line contains n integers t_1, t_2, ..., t_n — how much time it takes Mitya to eat one cookie in vertex i ($1 \leq t_i \leq 10^6$).

Each of the following $n - 1$ lines describe the tree. For every i from 2 to n, the corresponding line contains two integers p_i and l_i, where p_i denotes the parent of vertex i and l_i denotes the time it takes Mitya to move the chip along the edge from vertex i to its parent ($1 \leq p_i < i$, $0 \leq l_i \leq 10^9$).

Output

Output a single integer — maximum number of cookies Mitya can eat.

Examples

input
5 26 1 5 1 7 7 1 3 2 2 2 1 1 1 1 2 0 2 0
output
11

input
3 179 2 2 1 6 6 6 1 3 2 3
output
4

Note

In the first example test case, Mitya can start by moving the chip to vertex 2. In this case no matter how Vasya plays, Mitya is able to eat at least 11 cookies. Below you can find the detailed description of the moves:

1. Mitya moves chip to vertex 2.
2. Vasya removes edge to vertex 4.
3. Mitya moves chip to vertex 5.
4. Since vertex 5 has no children, Vasya does not remove any edges.
5. Mitya stops the game and moves the chip towards the root, eating cookies along the way (7 in vertex 5, 3 in vertex 2, 1 in vertex 1).

Mitya spend $1 + 0$ time to go down, $0 + 1$ to go up, $7 \cdot 2$ to eat 7 cookies in vertex 5, $3 \cdot 3$ to eat 3 cookies in vertex 2, $1 \cdot 1$ to eat 1 cookie in vertex 1. Total time is $1 + 0 + 0 + 1 + 7 \cdot 2 + 3 \cdot 3 + 1 \cdot 1 = 26$.
