Problem A. Black Frame

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

When it's freezing outside, there isn't much to do in Waterloo. Luckily for you, you stumbled onto the "Black Square" painting by Kazimir Malevich. While that painting is literally just a black square, you were marveled by its simplicity and use of color. And that's how your weekend art project was born. You took some old jigsaw puzzle and painted every single puzzle piece black. As a result, you ended up with pieces of seven kinds, as shown in the picture below. You have varying quantities of each kind. By interlocking puzzle pieces, you connect them together. The "Black Square" was already painted and you're interested in making frames for it, which also must be completely black. Since you're also interested in combinatorics, the following question entered your mind.

You're interested in how many integers $n \geq 3$ exist such that you can make an n by n frame from the jigsaw puzzle pieces that you have (each puzzle piece is 1 by 1). You don't have to use every single puzzle piece that you have. You can rotate puzzle pieces but you are not allowed to flip them over (as the back texture is not the same as the front texture). An n by n square frame is a structure of size n by n whose $n-2$ by $n-2$ interior is completely empty. That means that you cannot have any puzzle parts sticking in or out, nor can you have any emptiness left in a puzzle piece. Lastly, each puzzle piece has to be connected to exactly two other puzzle pieces. This means that putting two flat sides together does not count as a valid connection.

Input

The only line of input contains seven integers ranging from 0 to 10^9 (inclusive) - the quantities of each puzzle piece kind. The order of input corresponds to the order shown in the image (so the first integer is the count of the leftmost puzzle piece kind, and so on).

Output

Output one number - how many valid values of n exist.

Examples

standard input	standard output
0 0 4 0 4 0 2	1
1 1 1 1 1 1 1	0
0 4 0 4 4 4 12	5

Problem B. Orb Tent

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

You have found a collection of N floating orbs of light, some of which are nice to look at, and some of which hurt to look at. We model this by each orb having a beauty value which is an arbitrary (possibly negative) integer. We want to build a tent which will house some subset of the orbs, and want the sum of the beauty of the orbs in the tent to be as large as possible. However, the tent must be convex.

Specifically, the position of each orb is at coordinates (x, y), where $0 < x < L$ and $y > 0$, and the tent must be a convex polygon that includes the points $(0, 0)$ and $(L, 0)$. Find the maximum sum of beauty of orbs inside any valid tent configuration.

An orb is considered inside the tent if it is exactly on the tent wall. The tent is considered convex even if some orb is exactly on a straight section of the tent wall (i.e. the angle of the tent wall at that orb is exactly 180 degrees). It is allowed for the tent to not contain any of the orbs (i.e. and just lie on the ground).

Input

The first line of input contains two integers $0 < N \leq 2000$ and $0 < L \leq 10^9$.

N lines of input follow containing three integers each: the $0 < x < L$ and $0 < y \leq 10^9$ coordinates of an orb and the beauty value $-10^6 \leq b \leq 10^6$ of that orb.

Output

Output a line containing a single integer, the maximum sum of beauty of orbs inside any valid tent configuration.

Example

standard input	standard output
10 8 2 7 -1 2 5 2 3 5 1 4 5 -10 6 5 4 6 2 -1 7 2 2 7 3 -1 4 3 3 1 3 1	9

Problem C. Whitewater Rafting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Whitewater rafting and canoeing are fun activities, especially on hot summer days. Splashing water cools you off in the rapids, then the current carries you to another part of the river.

There are many different rivers to choose from, each with its own set of rapids. Which river is the most fun, and which section of that river?

From each rapid i, the current flows to some unique other rapid i', except when i is the last rapid on a given river. It is possible for one river to flow into another; in this case, the current flows from two or more different rapids $i_1, i_2, ...$ to the same downriver rapid i'. Since water always flows downhill, it is not possible for the current to start at some rapid i and eventually return to the original rapid i.

To make fun precise, we will assign each rapid i a score $-1000 \leq s_i \leq 1000$. You can choose to start your trip at some rapid, then flow with the current to successive rapids until you either get tired and decide to stop, or until you reach the last rapid of a river. The overall score of a trip is the sum of the scores of the rapids that you passed through. You are at some starting rapid and wondering where you should stop to maximize the overall score of your trip.

Sadly, as summer turns to fall, water levels get lower and some sections of the river are no longer navigable. In other words, where previously the current flowed from rapid i to rapid i', it now no longer does. It is still possible to start a trip at rapid i', but any trip that goes to rapid i must stop there; it can no longer continue to rapid i'.

Input

The first line contains two integers - N and Q, with $1 \leq N, Q \leq 300,000$.

This is followed by N lines that describe the initial state of the rivers. The i'th such line contains two integers p_i ($0 \leq p_i \leq N$) and s_i ($-1000 \leq s_i \leq 1000$). If p_i is 0, then rapid i is at the end of a river. Otherwise, p_i indicates the next downriver rapid that rapid i flows into. s_i is the fun score of the i'th rapid.

This is followed by Q lines that describe changes to the water levels over time and queries. Each of these Q lines contains two integers c ($1 \leq c \leq 2$), x ($1 \leq x \leq N$). If c is 1, then this line indicates that rapid x now becomes the end of a river (i.e. it no longer flows to any other rapid). In this case, it is guaranteed that rapid x was not already the end of a river. If c is 2, then this query asks you to output the maximum amount of fun that can be obtained by starting your trip at rapid x.

Output

Output one line for each query with $c = 2$, indicating the maximum fun you can have for that query. Output should be in the same order as the queries appear in the input.

Examples

standard input	standard output
2 3 0 5 1 5 2 2 1 2 2 2	10 5
5 5 0 5 1 -1 2 5 0 3 2 0 2 3 1 2 2 3 2 4 2 2	9 5 3 -1

Problem D. Reptilian Communism

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There are three types of chameleons inhabiting New Waterloo Island - red, green, and blue. When two chameleons of different colors meet, they both change their color to the third color and breed k new chameleons of that color. For example, when red and blue chameleons meet, they both become green as a result of this encounter and k new green chameleons appear. When two chameleons of the same color meet, nothing happens. When every single chameleon on the island becomes red, the communist revolution will happen on the island. Your task is to determine whether there exists a sequence of chameleon encounters that would lead to the communist revolution, and if such a sequence exists, give an example of one.

Input

The first line of input contains three integers r, g, b - the number of red, green, and blue chameleons. The second line of input contains a single integer k. $2 \leq r, g, b \leq 10^4, 1 \leq k \leq 10^4$

Output

Output `YES` if the communist revolution can happen, and `NO` otherwise. If you've outputted `YES`, in the next line print t - the number of encounters. In the t lines that follow, print two characters separated by a space, each of those characters being one of `R`, `G`, `B`. The symbols `R`, `G`, `B` correspond to red, green, and blue chameleons, and a string `R B` would mean a meeting between red and blue chameleons. If there are several ways to set up these encounters, print any one of them. However, the largest allowed t is 10^5 - beyond that, communist chameleons are going to give up their cause.

Examples

standard input	standard output
3 2 2 2	YES 2 B G B G
17 7 4 3	NO

Note

In the first test, we're setting up two encounters between green and blue chameleons. The quantities of chameleons are going to change as follows: 3,2,2 -> 7,1,1 -> 11,0,0. The revolution will happen!

Problem E. Loose Goose

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

Competitive programming is awesome, but bartending can be a better way to make some extra cash. Tonight you're bartending at the "Lonely Goose" bar at Waterloo, serving their signature "Loose Goose" cocktail. There are n glasses lined up in a row at a bar table. Initially, all glasses are empty and the goal is to have a_i millilitres of "Loose Goose" in the i-th glass. You're a skilled bartender, and each minute you can choose a contiguous set of glasses and pour either one millilitre of cocktail into each glass or x millilitres into each glass. It is forbidden to pour out excess liquid from a glass (which would be wasteful). Find out the quickest time in which you can achieve the desired drink allocation.

Input

The first row contains two integers - n and x. The next row contains n integers a_i.

$1 \leq n \leq 300000, 2 \leq x \leq 10^9, 0 \leq a_i \leq 10^9$

Output

Output a single number - the smallest possible number of minutes to achieve the goal.

Examples

standard input	standard output
6 3 1 1 1 4 3 3	2
5 5 4 0 4 0 4	12

Note

In the first test, you can pour 1 millilitre into glasses 1-4, followed by pouring 3 millilitres into glasses 4-6.