Problem A: Easiest Problem

Filename: `easiest`
Time limit: `3 seconds`

Arup was worried nobody would solve a problem during Contest 4, so he asked Spencer to make an extremely easy problem. Spencer made the problem, but when Jacob tried test-solving it, he was able to solve it in a couple of seconds. Spencer wondered, is this problem too easy for an SI contest? Probably, but it was too late to think of another problem.

Can you solve the problem Jacob could? The problem is as follows:

Farmer John has ***N*** cows currently in ***N*** different barns connected by ***N-1*** trails. It is guaranteed there exists exactly one path from every barn to every other barn using one or more of these trails. Each cow is one of two breeds: Guernsey or Holstein. Guernseys do not like Holsteins, and vice versa. Two cows will be friends if they are the same breed, are at most distance ***K*** apart, and the path between them contains no cows of the opposite breed. The distance between two friends is the number of trails on the path between them. Help Farmer John calculate the number of valid pairs of friends.

Input

The first line of input contains two positive integers: ***N*** ($1 \le N \le 10^5$) and ***K*** ($1 \le K \le 10^5$). The second input line will contain ***N*** space separated characters (either H or G) representing the breeds of cows, in order. The ***i***th line of the following ***N-1*** lines each contain two distinct integers, A_i and B_i ($1 \le A_i, B_i \le N$, $A_i \ne B_i$), indicating that barns A_i and B_i are connected by a trail.

Output

Print out the number of valid pairs of friends.

Samples

Input	Output

`7 1`
`G H H H G G H`
`1 3`
`5 6`
`2 7`
`2 3`
`4 3`
`5 3`
	`4`

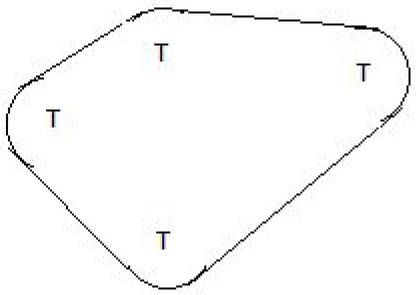
`6 2`
`G H G G G G`
`1 2`
`1 3`
`1 4`
`1 5`
`1 6`
	`10`

Problem B: Farmer John's Forest

Filename: `forest`
Time limit: `1 second`

Farmer John recently got an opportunity to acquire a forest next to his many pastures. Naturally, some of his cows venture from the pastures to the forest, out of curiosity. Unfortunately, the forest isn't a friendly place for a cow. Many creatures, such as snakes, lurk and can cause harm to Farmer John's cows. To prevent the cows from getting into the forest, Farmer John has decided to build a fence.

He would like to build a single fence of minimum length that encloses all of the trees in the forest. For the purposes of this problem, each tree in the forest can be treated as a point on the Cartesian plane, but there must be at least a ***c*** foot distance from any tree to the fence, so that the fence does not interfere with the growth of the roots of the tree. Here is an illustration of a fence with a minimum clearance for each of the four trees. The position of each tree is labeled with the letter 'T':



Write a program to help calculate the minimum length of fence that is necessary for Farmer John, as well as the area that will be enclosed by that fence.

Input

The first line of input will contain a two positive integers: ***n*** (***n*** ≤ 50,000), and ***c*** (***c*** ≤ 1000), representing the number of trees in Farmer John's Forest that need to be enclosed and the minimum distance of any tree to the fence, respectively. ***n*** lines of input follow. The ***i***th of these contain two space separated integers, x_i and y_i ($-10^4 \le x_i, y_i \le 10^4$), the Cartesian coordinates of the ***i***th tree.

Output

On a single line, output the minimum length of Farmer John's Fence, as well as the area it encloses. Output both values to precisely two decimal places and separate the output on the line with a single space.

Samples

Input	Output

5 1
0 0
10 0
0 10
5 5
10 10
	46.28 143.14

6 4
10 41
31 20
50 20
30 41
40 30
21 20
	126.84 971.59

Problem C: MASSIVE LEGEND HERE

Filename: `legend`
Time limit: `6 seconds`

Max is opening Pokémon cards for his YouTube channel! For this video, he purchased (in tremendous bulk) a number of Pokémon card booster packs. The booster packs are part of a set of several collectible promotional cards, each with a varying degree of rarity. Max knows how many cards are in the set as well as the individual rarity of each card.

Like any Pokémon master, Max wants to Catch 'Em All™! He doesn't want to disappoint his subscribers, so he requires a particular degree of certainty that he will open every single different card among all of the individual booster packs. Each booster pack contains a single card.

Given the individual rarities of each card and the level of certainty Max wishes for in his pack opening, find the minimum number of booster packs Max should open such that he can expect (with his requested certainty) that he will find every individual card.

Input

Each test case will begin with a single line containing an integer $\boldsymbol{N}$ ($1 \le \boldsymbol{N} \le 20$) and a decimal $\boldsymbol{C}$ ($0 < \boldsymbol{C} \le 0.99$), the number of distinct cards in the promotional set and Max's requested level of certainty. Each of the following $\boldsymbol{N}$ lines will contain two integers $\boldsymbol{p_i}$ and $\boldsymbol{q_i}$ ($1 \le \boldsymbol{p_i} \le \boldsymbol{q_i} \le 10^5$), indicating that the $\boldsymbol{i}$th card type occurs $\boldsymbol{p_i}$ times in every $\boldsymbol{q_i}$ packs. It is guaranteed that $\sum_{i=1}^{N} \frac{p_i}{q_i} = 1$.

Output

Print the minimum integer P such that, if Max purchases P booster packs, he has a probability of at least $\boldsymbol{C}$ of acquiring each individual card.

Samples

Input	Output

`3 0.3333`
`1 2`
`1 3`
`1 6`
	`4`

`5 0.895`
`1 8`
`1 8`
`1 8`
`3 8`
`1 4`
	`25`

Problem D: Productive Pastures

Filename: `pastures`
Time limit: `6 seconds`

Farmer John has N pastures on his farm. Ever the penny-pincher, FJ only constructed N - 1 paths between these pastures, taking care to ensure that he could get from any pasture to any other pasture using these paths. Each of these paths generates some amount of productivity by allowing cows to travel between the pastures they connect.

The one place that FJ didn't skimp was security - he installed retinal scanners at the main entrance for each pasture. However, in a twisted turn of events, all of the retinal scanners have gone down at once! (FJ was too cheap for regular maintenance). Luckily, he's got K technicians available to go fix the scanners.

Each technician can only fix one scanner today, and FJ wants to get his farm's productivity as high as possible as soon as he can. If he assigns the technicians to the appropriate pastures to repair the scanners, how much productivity can he restore? A path will only be productive if the pastures on both ends of it have been visited by a technician.

Input

The first line of input contains two positive integers N ($N \leq 10^4$),and K ($K \leq 100$), indicating the number of pastures and the number of technicians respectively. The following N - 1 lines will each contain three integers u_i, v_i,($1 \leq u_i$, $v_i \leq$ N, $u_i \neq v_i$) and p_i ($1 \leq p_i \leq 10^6$), indicating that distinct pastures u_i and v_i are connected by a path which generates productivity p_i .

Output

Print a single integer, the maximum amount of productivity FJ can restore.

Samples

Input	Output

`6 3`
`5 3 3`
`2 4 5`
`1 3 2`
`4 6 3`
`3 4 4`
	`9`

`5 4`
`1 4 6`
`3 2 1`
`5 2 6`
`3 4 1`
	`12`

Problem E: Fire Sale 2

Filename: `firesale2`
Time limit: `2 seconds`

There is a fire sale going on at Anya's favorite toy store again! As Arup wants to make Anya happy, he gives her a maximum allowance of ***k*** dollars. It is Anya's goal to buy as many toys as possible without spending more than ***k*** dollars.

However, the fire sale has certain rules. All of the toys are lined up in a row, and Anya can only buy toys that are in one contiguous sequence. The store owner is mad because how much money he lost last time Anya came into his store. Now, the cost of the sequence of toys is the average of all the toys in the subarray.

Input

The first line of input will contain two space separated integers, ***n*** ($1 \le n \le 10^6$), ***k*** ($1 \le k \le 10^6$) representing the number of toys in the store and the maximum amount of money Arup is willing to spend. On the following line is ***n*** space separated integers, a_i ($1 \le a_i \le 10^6$), the costs of each toy.

Output

On a single line, output the most toys Anya can buy without going over her allowance.

Samples

Input	Output

`5 1`
`5 5 5 5 5`
	`0`

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

`6 4`
`1 1 6 9 8 10`
	`3`

Problem F: Dank Memes

Filename: `memes`
Time limit: `3 seconds`

As you well know, one big problem in the land of dank memes is stolen memes. However, to impress your normie friends you've decided that you're going to start stealing dank memes to send to them. Because you're quite the meme lord, you have a vast selection of memes to pick from. In order to keep up your facade of being original, you want to minimize the chance of getting caught with stolen memes.

The $\boldsymbol{i}^{th}$ meme in your meme library has two associated values: $\boldsymbol{r_i}$, how recognizable the meme is, and $\boldsymbol{d_i}$, how dank the meme is. To gain the most meme clout with your pals, you want to choose dank memes such that their total dankness is maximal. You also want to pick a set of stolen memes with total recognizability no larger than $\boldsymbol{R}$. The total recognizability of a set of memes is simply the sum of the recognizability scores of each of those memes.

Given $\boldsymbol{R}$, a set of $\boldsymbol{n}$ stolen memes, each with their respective $\boldsymbol{r_i}$ and $\boldsymbol{d_i}$ values, determine the largest total dankness of memes you can share with your normie friends. Remember, you can't use the same meme twice, because then you're just another normie.

Input

The first line of input contains two integers $\boldsymbol{n}$ and $\boldsymbol{R}$ ($1 \le \boldsymbol{n}, \boldsymbol{R} \le 4{,}000$). Line $\boldsymbol{i}$ of the next $\boldsymbol{n}$ lines describes the $\boldsymbol{i}^{th}$ stolen meme with two integers $\boldsymbol{r_i}$ and $\boldsymbol{d_i}$ ($1 \le r_i \le 3{,}000$; $0 \le \boldsymbol{d_i} \le 10^9$), denoting the recognizability and dankness respectively.

Output

Print out a single integer $\boldsymbol{D}$, the largest total dankness you can achieve by sharing some set of stolen memes with your friends without surpassing $\boldsymbol{R}$ in recognizability.

Samples

Input	Output

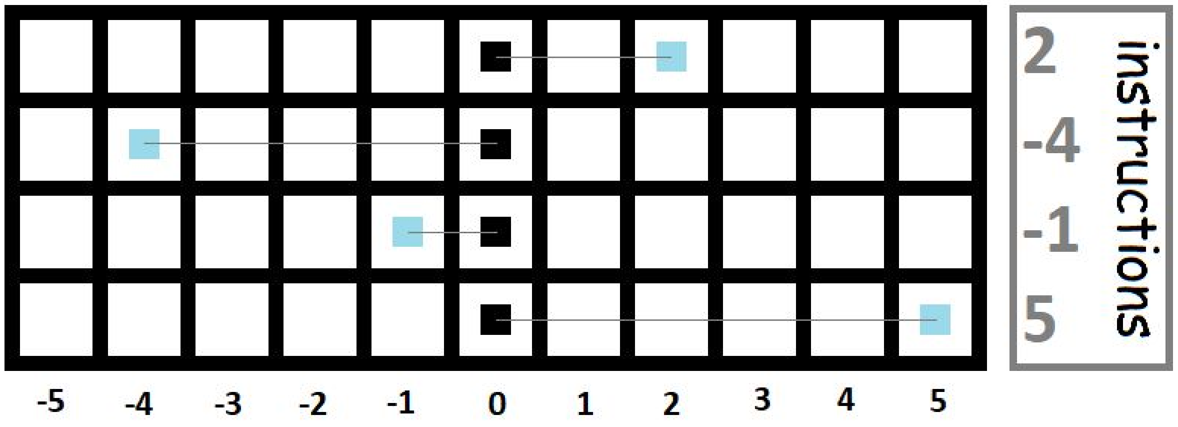
`5 20`
`1 2`
`2 1`
`3 4`
`4 3`
`5 5`
	`15`

`3 20`
`20 1000`
`19 999`
`1 1`
	`1000`

Problem G: Ballerina Rage

Filename: `rage`
Time limit: `3 seconds`

The International Ballerina Monarchy has decreed that all of its citizens must stand on a grid and occupy a variety of locations! The Monarchy can instruct its citizens to move left and right using a variety of ***C*** instructions cards the cardinals prepared earlier. The ***i***th number on each card serves as an instruction for the ***i***th citizen in line on the grid to move a certain distance. Look below for an example instruction card and the effect it has if each citizen begins at location



0.
There are ***N*** citizens, and ***N*** infinitely long horizontal rows for them to stand on. Each of the citizens start at location 0 in their respective row. The Monarchy has ***L*** positions they would like the citizens to move into, and they would like to know whether or not they can use the instruction cards to get them there.

Note that all of the cards can be used entirely forward or entirely reverse. For example, if N = 3, and the citizens are currently at (1, 1, 1), an instruction set of (1, -2, 4) could be used to bring the citizens to either (2, -1, 5) or (0, 3, -3). Furthermore, any fraction of the card's values can be used as well. An instruction set of (3, 5, 7) could be scaled down and used like (0.6, 1, 1.4). The cards can also be reused as many times as desired.

Input

Each case will begin with a single line containing three integers ***N*** (1 ≤ ***N*** ≤ 50), ***C*** (1 ≤ ***C*** ≤ 50), and ***L*** (1 ≤ ***L*** ≤ 100), corresponding to the number of citizens, the number of instruction cards, and the number of locations the Monarchy has chosen. ***C*** lines will follow, each will contain ***N*** integers a_i ($-10^4 \le a_i \le 10^4$), indicating that the ***i***th number on the instruction card is a_i. Then ***L*** lines will follow, each containing ***N*** integers p_i ($-10^9 \le p_i \le 10^9$), indicating that the Monarchy would like citizen ***i*** standing at location p_i for all i simultaneously.

Output

For each of the ***L*** sets of positions, output “YES” on a line by itself if the Monarchy can use the cards to get the citizens to that set of locations, or “NO”, otherwise.

Samples

Input	Output
`3 3 2`	`YES`
`1 1 0`	`YES`
`4 -3 -1`	
`0 -9 7`	
`8 7 -3`	
`-2 1 5`	
`4 3 4`	`NO`
`1 2 3 1`	`YES`
`1 2 3 2`	`YES`
`1 2 3 3`	`NO`
`3 4 1 5`	
`6 12 18 6`	
`1 2 3 1`	
`5 4 9 11`	

Problem H: Get Spencer Off Campus

Filename: `spencer`
Time limit: `1 second`

Spencer is brilliant when it comes to algorithms but terrible when it comes to directions. He's been having trouble navigating his way around campus. So far, Charles has been helping him find the lecture room, or Arup has been letting him just stay in HEC-101.

But this time, neither Arup nor Charles are around. You must help Spencer get off campus. If he can't leave UCF, Spencer won't be able to earn a Gold Medal at IOI or attend MIT for that matter! This project is a matter of national security. Don't let Spencer down!

We model campus as a ***R*** x ***C*** grid of squares. Spencer will initially occupy a single one of those squares. At any moment in time, Spencer can be facing one of four directions, up, down, left or right. Some of the campus grid squares are obstructed and Spencer can not travel to those squares. Otherwise, Spencer can move in a straight line in his current direction as many squares as possible until there is an obstructed square, without any trouble. Alternatively, Spencer can turn either left or right, which results in him staying in the same grid square, but changing his direction. (If he wanted to do a u-turn, he would have to either turn left twice or turn right twice.)

The goal of this problem is to get Spencer off campus. Since Spencer has extreme difficulty following directions with many turns, the goal will be to get Spencer to any unobstructed border square (in either the first or last row, or first or last column) ***using as few turns as possible.***

Write a program to calculate the fewest number of turns necessary to get Spencer off campus, so that he can bring glory back to the United States and attend his intended college. If this is not possible, your program must report that the task is impossible.

Input

The first line of input has two integers ***R*** ***(***3 ≤ ***R*** ≤ 50) and ***C*** (3 ≤ ***C*** ≤ 50), representing the number of rows and columns, respectively, on the grid. The following ***R*** rows have a string of precisely ***C*** characters. Each of these characters will either be '_' , 'X', 'N', 'S', 'W', or 'E'. The character '_' indicates a passable square. The character 'X' indicates an obstructed square where Spencer can not travel. Of the four characters 'N', 'S', 'W' and 'E', exactly one of them will appear in the grid and that character will appear exactly one time. The location of this character represents Spencer's initial starting square. The letter itself represents which direction Spencer is initially facing, 'N' for North, 'S' for South, 'W' for west, and 'E' for east. It is guaranteed that at least one square on the border of the board will not be an obstructed square.

Output

On a line by itself, output a single integer, indicating the fewest number of ***turns*** necessary for Spencer to get off campus (to a border square). If it’s not possible for Spencer to do so, output -1.

Samples

Input	Output
5 5 XXX__ XNX_X X_X_X X___X XXXXX	4
5 7 XXXXXXX __XXX__ __XWX__ __XXX__ _______	-1

Sample Explanation

In the first example, Spencer must turn either left or right twice, so that he’s facing South. Then he will travel south two squares before turning left. Then he will travel two more squares east before turning left again. From here, Spencer can go straight and get off campus to row 0, column 3.

Problem I: Byteland Bands Together

Filename: `together`

Time limit: `2 seconds`

Byteland is a very long rectangular land comprising of ***n*** different states arranged as contiguous squares in a line. In order, these states are labeled 1 to ***n*****,** from left to right. Historically, each state had its own constitution, but the country has banded together to form a single unified constitution!

In order to agree on a single constitution, the process to be used is as follows:

If the contiguous states from state ***i*** to ***j*** (***i*** ≤ ***j***) share a constitution and the contiguous states from ***j***+1 to ***k*** (***j*** < ***k***) share a different constitution, then the delegates from both groups, [***i***, ***j***] and [***j***+1, ***k***] can meet for a convention in either state ***j*** or state ***j***+1 to hammer out the differences and agree on a single unified constitution for the contiguous states in the range [***i***, ***k***]. Unfortunately, hosting such a convention costs money. In particular, in order to entice each of the delegates to attend the convention, the planners must buy each of them a famous Byteland Taco! Due to differences in economies of the different states, the cost of the famous Byteland Taco varies from state to state. In particular, in state ***i***, the cost of a taco is c_i. Also, the different delegations from each state vary in size. The delegation from state ***i*** has m_i members.Thus, the total cost for holding the convention at state ***j*** is $c_j \sum_{p=i}^{k} m_p$.

Over the course of ***n-1*** conventions, Byteland will have its unified Constitution. Unfortunately, depending on the order of these conventions, the total cost of unifications may vary significantly. Although tacos are delicious, Byteland would like to minimize the cost of the tacos over all of the conventions.

Given the cost of tacos in each state and the number of delegates from each state in Byteland, determine the minimum cost of tacos required for the country to agree on a single Constitution.

Input

The first line of input contains a single positive integer ***n*** (1 ≤ ***n*** ≤ 500), representing the number of states in Byteland. The next line of input contains ***n*** integers, the ***i***th of which is m_i ($1 \le m_i \le 10^5$), the number of members in the delegation from state ***i***. The last line of input has ***n*** integers, the ***i******th*** of which is c_i ($1 \le c_i \le 10^5$), the cost of a taco in state ***i***.

Output

Output the minimum cost of tacos necessary for the conventions for Byteland to agree on a single constitution.

Samples

Input	Output

4
1 9 6 3
3 6 9 1
	145

2
1 6
9 3
	21