Problem A: Bunny Farm

Filename: `bunny`
Time limit: `2 seconds`

Farmer Eric runs a bunny farm where he has ***n*** bunny dens in a row. He wants to do an analysis on the population growth of bunnies. When two groups of bunnies merge, he knows that the population of the resulting den is the product of the two groups. However, he is not very good at arithmetic with large numbers.

Farmer Eric can perform a single operation and would like to be able to perform a single query periodically. The action is merging a new den with an existing den, increasing the total population size of that den (to their product). The type of query he'd like to perform is as follows: given two groups if he merged group 1 into one den, and group 2 into another den, which group would have the larger population? A group is defined by two integers, ***l*** and ***r***, the start and end index, inclusive, of the group from the ***n*** dens.

Input

The first line of input will contain two space separated integers, ***n*** ($1 \le n \le 10^5$), and ***q*** ($1 \le q \le 10^5$) representing the size of the farm and the number of actions Farmer Eric will take. The next line contains ***n*** integers, a_i ($1 \le a_i \le 10^{18}$, a_i is guaranteed to be a power of 2), representing the population of each bunny den. The following ***q*** lines will each contain information about one action. The i^{th} of these lines will contain ***t*** ($1 \le t \le 2$), the action type.

For action type 1, there will be two space separated integers, ***j*** ($1 \le j \le n$), and ***x*** ($1 \le x \le 10^{18}$, ***x*** is guaranteed to be a power of 2), the index of the bunny den and the population size of the new den being merged into the j^{th} bunny den.

For action type 2, there will be four space separated integers, l_1, r_1, l_2, r_2 ($1 \le l_1, r_1, l_2, r_2 \le n$) the start and end indices of group 1 and group 2. It is guaranteed that $l_1 \le r_1$ and $l_2 \le r_2$.

Output

For each action of type 2, print 1 if group 1 has the larger population product, -1 if group 2 has the larger population product, or 0 if they are equal.

Samples

Input	Output
5 5 1 2 4 8 16 2 5 5 4 4 1 4 2 2 5 5 4 4 1 4 2	1 0 -1

2 5 5 4 4	

Problem B: Favorite Divisors

Filename: `divisors`
Time Limit: `1 second`

Farmer John has two favorite numbers, ***N*** and ***K***. He also considers a number *X* one of his favorite divisors if *X* is a divisor of ***N*** and *X* has exactly *K* divisors (including 1 and *X*). Bessie would like to impress Farmer John by figuring out how many favorite divisors he has. Can you help her?

Input

The first line of input contains two space separated positive integers: ***L*** ($1 \leq L \leq 200$), and ***K*** ($1 \leq K \leq 10^{18}$). The second input line will contain ***L*** space separated positive integers less than 10^9. ***N*** ($1 \leq N \leq 10^{42}$) is equal to the product of all integers on the second line.

Output

Print out the number of favorite divisors Farmer John has.

Samples

Input	Output
3 4 5 2 3	3

Sample Explanations:

K is equal to 4 and *N* is equal to the product of 2, 3, and 5. There are 3 divisors of 30 that have exactly 4 divisors: 6, 10, and 15.

Problem C: Finicky Anya

Filename: `finicky`
Timelimit: `2 seconds`

When Arup is taking care of Anya, he is a softie and he doesn't want her to fuss or cry. So, to prevent her from crying, he will buy her something, typically a stuffed animal. As one might imagine, this gets expensive after a while. Arup would like your help in minimizing how much he spends while making sure Anya doesn't fuss.

For the purposes of this problem, assume that Arup and Anya are walking along a number line in the positive direction, starting at 0, ending at some fixed point *e*, and that initially Anya has some happiness score. Each minute, Anya's happiness score decreases by 1 and the two of them walk 1 unit. At various locations along the number line, there will be an opportunity for Arup to buy Anya an item. Each item has a cost and amount of increase in happiness it will bring Anya. If Anya's happiness ever goes down to 0, she will fuss. Note that if her happiness goes down to 0 and immediately Arup buys her an item at that instant, Arup (and Anya) will be okay.

Given Anya's initial happiness, the length of their walk, a list of locations along the number line with opportunities to buy items, the cost of those items AND the amount of added happiness those items will bring to Anya, determine the minimum amount Arup can spend to make sure that Anya doesn't get fussy for one or more minutes.

Input

The first line of input will contain three space separated integers, h ($1 \le h \le 100$), e ($2 \le e \le 1000$), and n ($1 \le n \le 100$) representing Anya's initial happiness, the ending coordinate on the number line for the walk, and the number of locations at which items are available for purchase, respectively. The following n lines will each contain information about one item for purchase. The i^{th} of these lines will contain three space separated integers, x_i ($0 < x_i < e$), c_i ($1 \le c_i \le 10000$), and a_i ($1 \le a_i \le 1000$), representing the x-coordinate the item is available for purchase (a positive integer), the cost of the item in cents, and the added happiness the item brings Anya at that point in time (a positive integer). The x_i's are such that $0 < x_1 < x_2 < x_3 < ... < x_n < e$.

Output

On single line, output the minimum Arup can spend in cents while making sure Anya does not cry for even one minute during their walk. If it's impossible to prevent Anya from crying, output -1.

Samples

Input	Output
15 11 1 3 99 5	0
5 20 1 7 1000 100	-1
5 20 3 2 559 4 4 899 20 9 339 11	898

Problem D: Cow Grouping

Filename: `grouping`
Time limit: `5 seconds`

All of Farmer John's cows are standing in a tremendously long line! Gazing off into the endless horizon of bovine beauty, an idea comes to FJ: all of the cows should get together for some group activities! But before the activities can begin, Farmer John must split his cows into groups.

To do this, FJ stands at the front of the line of cows. He begins by adding the first cow in the line to an empty group. There are many different breeds of cow, and Farmer John knows that the groups will get along well if they each have a strong and united leadership, so he continues to add cows to the group until there are K cows of a single breed within the group. (Note: at the point in time that an added cow is the K^{th} cow of her breed, the group is immediately finalized. This means that the group must have fewer than K cows of all other breeds.) Once a group is made, he sends them off to do some activities and then continues taking cows from the front of the line, building more and more groups.

Farmer John is quite certain that he wants K cows of a single breed within every group, so if he gets to the end of the line and the very last cow isn't the K^{th} cow of some breed in the current group, he won't be satisfied. However, he's not sure what value of K to use, or how many groups some K might result in.

Find all values of K that are satisfactory to Farmer John, as well as the number of groups each value of K splits the cows into.

Input

Each test case will begin with a single line containing two integers $\boldsymbol{N}$ ($1 \leq N \leq 3 \times 10^5$), and $\boldsymbol{M}$($1 \leq \boldsymbol{M} \leq 100$), indicating the number of cows and the number of cow breeds, respectively. The next line will contain $\boldsymbol{N}$ space separated integers $\boldsymbol{a_i}$ ($1 \leq \boldsymbol{a_i} \leq \boldsymbol{M}$). If $\boldsymbol{a_i} = \boldsymbol{j}$, then the i^{th} cow is of breed type j.

Output

On the first line print an integer P: the number of satisfactory options for K.

Then print P lines, each containing two integers k_i and g_i, indicating that if Farmer John breaks groups off when there are k_i cows of a single breed in the group, then exactly g_i groups will be formed. Print these lines in order of k_i, from smallest to largest.

Samples

Input	Output
5 2 1 2 2 1 1	3 1 5 2 2 3 1
7 3 1 2 3 1 2 3 2	3 1 7 2 2 3 1

Sample Explanation

In the first sample, this is what the groups would look like for each valid K:
1: [1] [2] [2] [1] [1]
2: [1, 2, 2] [1, 1]
3: [1, 2, 2, 1, 1]

Problem E: Anya and Ice Cream

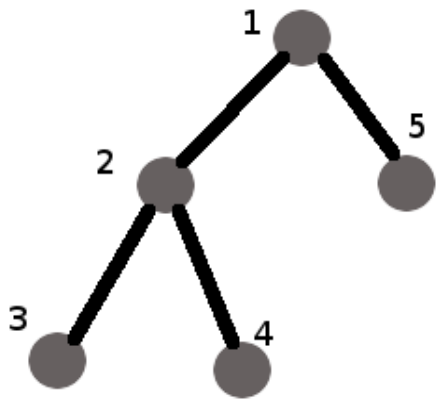
Filename: `icecream`
Time limit: `3 seconds`

Anya is quite smart. Recently, she graduated as valedictorian from Berland High. Her parents were very proud, so they gave her a special gift.

Because Anya hates cycles and non-connectedness, and loves ice cream, her parents gave her a tree rooted at node **1**. Each node in the tree has an ice cream tub with capacity k_i gallons. On the ***i***'th day of summer, Anya wants to pick some tub of ice cream from the ***i***'th subtree of the rooted tree such that the tub has at least ***K*** gallon capacity. If possible, she will eat all of the ice cream from that tub, responsibly wait twenty minutes, and then jog to the grocery store (to work off that ice cream) and buy enough ice cream to refill the tub she picked that day. However, Anya will be sad if on the ***i***th day, the ***i***th subtree in her spectacular ice cream tree has no such tub of at least ***K*** gallons. In fact, if Anya is sad just one day, her entire summer is a waste.

Help determine if Anya will have a good summer by answering ***Q*** queries.

Subtrees are numbered by traversing the tree from the top (also known as the preorder traversal). The root of the tree is the root of the 1^{st} subtree. Refer to the image on the right for an example. If two or more nodes have the same parent, the earlier subtree is the one rooted at the node with the smaller index ***i*** as indicated in the input.



Input

For each test case:

- The first line contains two integers, $1 \le N,Q \le 10^5$ the number of nodes in Anya's ice cream tree and how many queries will follow, respectively. The following line contains ***N*** integers, the i^{th} of which is the capacity $1 \le k_i \le 10^9$ of the i^{th} ($1 \le i \le N$) ice cream tub. Next is a line containing ***N - 1*** integers. The i^{th} ($1 \le i \le N-1$) of these integers is p_{i+1}, the parent of node ***i***+1.
- The following ***Q*** lines contain ***Q*** queries, one per line. Each query consists of two space separated integers ***i*** ($1 \le i \le N$) and ***K*** ($1 \le K \le 10^9$), the day of summer, and the minimum capacity tub Anya wants that day, respectively.

Output

For each query, output "YES" if Anya can get what she wants or output "NO", otherwise.

Samples

Input	Output
3 3 1 10 1 1 1 1 10 2 10 3 10	YES YES NO
5 2 1 2 3 4 5 1 2 3 4 5 2 2 5	YES YES

Sample Explanation

For the first sample, in the third query, Anya cannot find a bucket of capacity not less than ***10*** in the third subtree (consisting of only the third node).

In the second sample, Anya can find a value not less than ***2*** in the fifth subtree (which is just node ***5***). For the second query, she can find a single bucket with capacity at least ***5*** in the second subtree (which is rooted at ***2***).

Problem F: Number of Depth First Searches

Filename: `numdfs`
Timelimit: `2 seconds`

On the back of your Summer Institute T-shirt is a picture of a depth first search, with each vertex labeled in the order that it was visited on a depth first search from the top left vertex that forms the 'S' on the shirt. This got you to thinking, how many different orderings of the vertices, starting from that top left vertex form valid depth first searches on a graph. Write a program to satisfy your query!

Input

The first line of input contains three space separated positive integers: n ($1 \le n \le 10$), m($n-1 \le m \le n(n-1)/2$), and v ($1 \le v \le n$), representing the number of vertices in the graph, the number of edges in the graph, and the starting vertex for the depth first search, respectively.

The following m lines of input each contain a pair of integers. The i^{th} ($1 \le i \le m$) of these lines contains integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), indicating that vertices u_i and v_i are connected by an edge in the graph. It is guaranteed that each edge listed will be unique and that the graph itself is connected.

Output

On a single line by itself, output the number of valid depth first searches of the input graph that ***start at*** vertex v.

Samples

Input	Output
3 3 2 1 2 1 3 3 2	2
5 4 3 1 2 5 1 3 5 4 5	2

Problem G: Paper Route

Filename: `paper`
Time limit: `3 seconds`

A newspaper company has opened up shop in town. The town consists of N numbered locations and M roads connecting these locations. There are K newspaper distribution centers occupying the locations numbered 1 through K. The rest of the locations are houses, to which newspapers must be delivered.

Each distribution center i has an infinite number of couriers which cost c_i to move across a single road. The couriers can only deliver a single paper each; the cost of their trip is c_i times the number of roads between the distribution center and their destination. The couriers always take the shortest path from the distribution center to their destination.

Every house must get a newspaper. Assuming you assign couriers from your distribution centers optimally, what is the minimum total cost to get a newspaper to everybody?

Input

Each test case will begin with a single line containing three integers N, M, and K. ($1 \le N \le 10^3$, $N - 1 \le M \le 10^6$, $1 \le K \le \min(N, 100)$), indicating the number of nodes, the number of edges, and the number of distribution centers. The next line contains K integers c_1, c_2, …, c_K, indicating the costs of the couriers ($1 \le c_i \le 100$) for each distribution center i. The next M lines contain two integers u_i and v_i, indicating an edge between those two nodes.

Output

Print a single line containing the minimum total cost to deliver a newspaper to every house.

Samples

Input	Output
13 20 3 65 24 20 2 5 5 13 13 8 8 3 3 11 11 9 9 1 1 12 12 7 7 6 6 2 13 11 13 4 4 9 13 10 5 10 6 10 7 10 1 10 7 4	389

Problem H: Rambling Teaching Assistants

Filename: `rambling`
Time limit: `1 second`

After a day of watching online videos for SI@UCF Competitive Programming Camp, you noticed that some teaching assistants rambled a lot, and some rambled less. You realized that if you could watch the lectures faster, you could spend more time programming!

Given the the total duration of each teaching assistant's video lecture for a week, and each teaching assistant's ramble rate, calculate how much extra time you could spend programming if you watched the videos at speeds based off of each teaching assistant's ramble rate. For example, if a teaching assistant has a ramble rate of 1.0 and his lecture is 45 minutes long, the time you need to spend watching the lecture is 45 minutes; if the ramble rate is 2.0, then you can finish watching the lecture in 22.5 minutes. If the ramble rate is such that adjusting the video accordingly makes you spend ***more time*** watching, you must still watch the lecture at that new speed because your mother paid lots of money for this camp!!!

Input

The first line of input contains a single positive integer, N ($1 \leq N \leq 20$), denoting how many lectures to watch. The following N lines contain information about each lecture, one per line. The first item on each of these lines is the name of the teaching assistant giving the lecture, the second item on each of these lines is a positive real number, r ($r \leq 100$), the ramble rate of the teaching assistant giving the lecture, and the third item on each of these lines is a positive integer, d ($d \leq 300$), representing the length of the same lecture, in minutes. Each name is at most 20 letters long and each ramble rate will be given to at most two decimal places.

Output

Output a single real number rounded to two decimal places: the amount of extra time, in minutes, saved for programming if you watch the videos according to the given ramble rates, instead of at regular speed. If time is lost, please print out the corresponding negative number.

Samples

Input	Output
3 Guha 1 60 Bailey 2 60 Magnuson 0.9 75	21.67
1 Compton 0.5 60	-60.00

Problem I: Retrospective Rating

Filename: `rating`

Time Limit: `3 seconds`

Recently, Charles was looking at his Codeforces rating chart. He thought to himself, “what if I could have skipped up to ***k*** contests? How much higher could my rating be?”

Given a list of Charles’ rating deltas (changes), output how much higher Charles’ final rating could be if he had skipped up to ***k*** contests.

Input

The first line of input contains an integer, ***n*** ($1 \le n \le 5*10^5$), representing the number of rating changes to process, and an integer, ***k*** ($0 \le k \le n$), representing the maximum number of rating changes to ignore. The next line contains ***n*** integers all between -335 and 335, inclusive, representing each of the rating changes between contests.

Output

On a line by itself, output a single integer representing how much higher Charles’ rating could be if he could ignore up to ***k*** of his contest performances.

Samples

Input	Output
`5 1` `-5 10 20 30 40`	`5`
`5 0` `-5 10 20 30 40`	`0`
`4 3` `10 20 -2 -4`	`6`

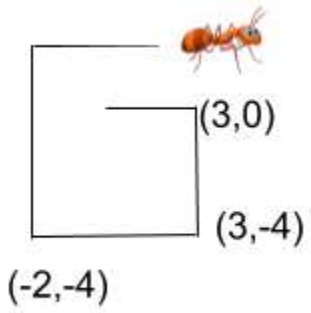
Sample Explanations

In the first sample, Charles can ignore one delta, so he ignores the first one. In the second sample, Charles cannot ignore any of the deltas. In the third sample, if Charles ignores a third delta, he score will go down, thus, his optimal choice is just to ignore the -2 and -4, even though he could have ignored a third.

Problem J: Spiraling Ant

Filename: `spiral`
Timelimit: `1 second`

Sammy the Ant lives on the Coordinate plane and loves to walk in spirals. On any given day, he simply chooses a positive integer ***x***, and then proceeds to start his walk from the origin. In particular, he will walk ***x*** units in the positive x-axis, then make a right turn and walk ***x*+1** units, then he'll make another right turn and walk ***x*+2** units, and so forth. After each walking segment, he always turns right and then travels in the new direction for one more unit than his previous walking segment.



(The diagram above shows Sammy's first few steps for when he chooses ***x*** = 3.)

Sammy is the keeper of the salami and you really love salami!!! Unfortunately, in order to get the salami, you have to find him. All you know is the value of ***x*** he chose for the day as well as how many straight line segments he walked for the day. Write a program to determine where Sammy is!

Input

The input consists of two positive integers, separated by a space: ***x*** (***x*** ≤ 100), and ***n*** (***n*** ≤ 10^5), representing the number of units of Sammy's first walking segment and the number of segments Sammy is walking for the day, respectively.

Output

On a single line by itself, output Sammy's x and y coordinate locations, respectively, separated by a space.

Samples

Input	Output
`3 4`	`-2 2`
`11 101`	`61 50`