Problem A: Cake for Angela

Filename: `cake`
Time limit: `1 second`

Recently, Donald Trump got into a bit of a tiff with German Chancellor Angela Merkel over various political issues. As it turns out though, Donald has reconsidered his position and would like to make up with her. Since today, July 17th, is Angela's birthday, he has decided that buying her a birthday cake will get him back in her good graces. In turns out that Angela is very picky about her cakes. Through secret knowledge acquired by the CIA, Donald has found out that Angela prefers a specific ratio between frosting and cake. The shape of her cake is that of a rectangular prism. It has a rectangle top and bottom and then is a certain height. Thus, the cake can be described to have a width, length, and a height. Unfortunately, Donald is not very good at math. Write a program to help him figure out how tall the cake should be, given its width and length, as well as Angela's desired ratio between frosting and cake. **The cake has frosting on the top and all sides. There is no frosting on the bottom of the cake. The frosting's thickness is negligible.**

Input

The first line of input contains two positive integers ***W*** and ***L*** (1 ≤ ***W***, ***L*** ≤ 100) representing the width and length of the cake in inches. The final line of input contains two positive integers ***A*** and ***B*** (1 ≤ ***A***, ***B*** ≤ 1,000). The fraction ***A*** / ***B*** is the desired ratio of square inches of frosting to cubic inches of cake. (The fraction ***A*** / ***B*** may not be in the most reduced form).

$$\frac{A}{B} = \frac{Square\ Inches\ of\ Frosting}{Cubic\ Inches\ of\ Cake}$$

Output

Output a single line containing two integers ***C*** and ***D*** representing the fraction ***C*** / ***D*** which is the height the cake should be to create the ratio of frosting to cake exactly equal to ***A*** / ***B***. Output the fraction in its most simplified and reduced form. It is guaranteed there will always be a valid solution (A positive height will always exist such that the optimal frosting/cake ratio can be achieved).

Samples

Input	Output
`12 12` `1 1`	`3 2`
`12 24` `5 9`	`36 11`

Problem B: Camper Grouping

Filename: `camper`
Time limit: `1 second`

When we were all at Universal on Saturday, the first logistical challenge was matching a group of campers with one or more adult figures such that both the campers and adults within a group shared the same level of interest in roller coasters.

Arup was thinking of adding more campers next year, and along with that he needs to add more adults to look after the kids. Arup wants to make sure that there are enough TAs in the camp such that: (a) the ith TA is responsible for no more than c campers at Universal; and (b) each TA is only paired up with campers sharing the same roller coaster interest level.

Help Arup check if there are enough TAs of each interest level hired.

Input

The first line of input has two integers n ($1 \le n \le 10^5$) and c ($1 \le c \le 10^9$), the number of interest levels and maximum number of campers a TA can supervise, respectively.

The following line contains n integers, the i^{th} of which is t_i ($1 \le t_i \le 10^9$), how many TAs have interest level i ($1 \le i \le n$). Another line follows with n integers, the i^{th} of which is d_i ($1 \le d_i \le 10^9$), how many campers have interest level i ($1 \le i \le n$).

Output

For each interest group, print a `1` if there are enough TAs to look after all of the campers of the same interest group. Otherwise, print a `0`. Print all of the `1`'s and `0`'s on one line with no spaces in between. Print a newline at the end.

Samples

Input	Output

`3 4`
`3 3 3`
`12 12 12`
	`111`

`3 4`
`3 3 3`
`13 12 11`
	`011`

Problem C: Stay Hydrated

Filename: `hydrated`
Time limit: `1 second`

On a hot day at Universal, it's important to stay hydrated! Luckily, most restaurants will give you a cup of water for free if you ask for it. For every hour spent walking in the sun, you should drink another 10 cups of water. Over time, given how much water you've drank in cups and how many hours you've been walking in the sun, calculate how many cups you still have to drink.

Input

The first line of input contains a single positive integer ***N*** ($1 \le N \le 9$) representing how many hours you've been walking in the sun. The next line contains ***N*** integers in the range [0, 20] for each hour, each representing how many cups of water you drank from some nearby "hydration station" during that hour.

Output

Output a single line containing ***N*** integers, each representing how many more cups of water you would still need to drink at the end of that hour to get hydrated.

Samples

Input	Output
`3` `5 10 2`	`5 5 13`
`3` `20 0 0`	`0 0 10`

Problem D: Get the Queen Out

Filename: `queen`
Time limit: `1 second`

In the game of Chess, the Queen is the most powerful piece. The game is played on an 8 x 8 board of squares. It can move any number of squares in any of the eight directions (up & left, up, up & right, left, right, down & left, down, and down & right), so long as there are no other pieces in the path.

In this problem, the Queen is placed on an arbitrary R x C board with some obstructed squares. The goal is to get her to the boundary of the board (either the first or last row, or first or last column) in as few moves as possible.

Given the arrangement of the board (where the queen is and which squares are obstructed), determine the fewest number of moves necessary to move the queen to any square on the boundary of the board.

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 board. The following ***R*** rows have a string of precisely ***C*** characters. Each of these characters will either be '_' , 'Q', or 'X'. The character '_' indicates a passable square. The character 'Q', of which there will be precisely one, indicates the initial position of the queen. The character 'X' indicates an obstructed square where the queen can not travel. 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 moves necessary for the queen to get to a border square. If it's not possible for the queen to do so, output -1.

Samples

Input	Output
5 5 XXXX_ XQX_X X_X_X X___X XXXXX	4
5 7 XXXXXXX __XXX__ __XQX__ __XXX__ _______	-1

Problem E: Airport Shuttle

Filename: `airport`
Time limit: `2 seconds`

When all of the out of state campers arrived to Orlando International Airport, the SI@UCF staff had to make several runs to the airport to pick up all of the campers. Naturally, none of the staff members wanted to wait too long at the airport. Each staffer's wait time was simply the difference in arrival times between the first and last camper he/she picked up.

Luckily, all of the staffers have access to arbitrarily large shuttle buses! But, in addition to wanting to minimize their wait time at the airport, none of the staffers want to make more than one airport run.

Given the number of SI@UCF staffers, as well as the arrival times of each camper flying into Orlando International Airport, determine the minimum amount of time, T, in minutes, such that no staffer will have to wait more than T minutes.

Input

The first line of input contains two space separated positive integers: ***n*** (***n*** $\leq 10^5$), and ***k*** (***k*** $\leq$ ***n***), where ***n*** represents the number of campers flying into the airport and ***k*** represents the number of staffers. The second line of input contains the ***n*** arrival times, in minutes, separated by spaces, of each camper. Each of these arrival times will be positive integers less than or equal to 10^9.

Output

On a single line by itself, output the minimum number of minutes T, such that no staffer waits more than T minutes, no staffer takes more than one trip, and each camper gets picked up.

Samples

Input	Output
`9 3` `10 10 30 200 205 210 215 220 500`	`20`
`5 2` `10 2 16 19 5`	`8`

Problem F: Brian and Castle

Filename: `castle`
Time limit: `4 seconds`

Brian has been playing Minecraft for so long that he fell asleep at his keyboard. Now he is dreaming about building a Minecraft castle, and for some reason you're there to help him. Brian has built several towers around his world, and he wants you to build some bridges between them so that he can reach any tower from any other tower using only the bridges.

However, because Brian is a harsh taskmaster with very specific wishes, you can not build bridges between any pair of towers you want. Brian has given you a list of pairs of towers you can build between, as well as how long he thinks it would take to build a bridge between those towers. You want to spend as little time as possible building these bridges so you can get out of this dream and go back to practicing programming problems. Brian guarantees that there is at least one set of bridges you can choose to satisfy his demands. How quickly can you finish building Brian's bridges?

Input

The first line of input contains two integers $\boldsymbol{N}$ ($2 \le \boldsymbol{N} \le 10^5$) and $\boldsymbol{M}$, ($\boldsymbol{N}$-1 $\le \boldsymbol{M} \le 10^5$), the number of towers and the number of possible bridges, respectively. The following $\boldsymbol{M}$ lines each contain three integers, $\boldsymbol{u_i}$, $\boldsymbol{v_i}$ ($1 \le \boldsymbol{u_i}, \boldsymbol{v_i} \le \boldsymbol{N}$, $\boldsymbol{u_i} \ne \boldsymbol{v_i}$), and $\boldsymbol{t_i}$ ($1 \le \boldsymbol{t_i} \le 10^6$), indicating that you can build a bridge between towers $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$ in $\boldsymbol{t_i}$ minutes.

Output

On a line by itself, print a single number, the minimum number of minutes it will take you to connect all of the towers in Brian's castle.

Samples

Input	Output
3 3 1 2 5 2 3 7 1 3 4	9
4 5 1 2 6 2 3 8 3 4 2 1 4 5 2 4 7	13

Problem G: RPS

Filename: `rps`
Time limit: `1 second`

Eric is playing rock-paper-scissors. However, Eric does not like to lose. Calculate the probability of winning ***n*** games of rock-paper-scissors without losing. Ties do not count as a loss.

Input

The first and only line contains an integer ***n*** ($1 \le n \le 10^{18}$) the number of games to win without losses.

Output

The probability of winning ***n*** games without any losses can be represented as a fraction reduced to lowest terms, p/q, where p and q share no common factors (gcd(p, q) = 1). Output the remainder when p+q is divided by 1,000,000,007.

Samples

Input	Output
`1000`	`688423211`
`10000`	`905611806`

Problem H: Toll Roads

Filename: `tollroads`
Time limit: `4 seconds`

Eric is trying to get from his house to campus as fast as possible, but he hates paying for tolls. When tolls were first introduced, there was promise of them being removed once the cost of the road was collected. Eric doubts this will ever happen, so he wants to try and cheat the system and is only willing to pay up to X dollars in tolls in one trip.

Given a description of the city, where each road takes time t_i and cost c_i in tolls (potentially zero if there is no toll on this road), help Eric find the shortest amount of time needed to get from his house (node 1) to campus (node N). Each road takes the same amount of time and cost no matter which direction the road is traveled along.

Input

The first line of input contains three integers N, M, and X, ($1 \le N, M, X \le 10^3$), the number of nodes in the city, the number of roads, and the maximum amount of dollars Eric is willing to pay in tolls.

The following M lines contain four integers each, u_i, v_i ($1 \le u_i, v_i \le N$, $u_i \neq v_i$), c_i ($0 \le c_i \le 10^5$) and t_i ($0 \le t_i \le 10^5$), describing the start and end of each road as well as the cost (toll) for the road and the time it takes to travel along the road.

Output

Print a single line containing the fastest time Eric can get to campus without overspending. If it is not possible for Eric to get to campus without overspending, print -1.

Samples

Input	Output

`4 6 4`
`1 2 2 100`
`1 3 2 200`
`1 4 5 1`
`2 3 2 5`
`2 4 2 50`
`3 4 2 20`
	`150`

`2 1 1` `1 2 2 1`	`-1`