3rd Annual SI@UCF Programming Competition Camp

Contest #1
July 10, 2017

Problems

Letter	Problem Name	Filename
A	Ant Clans	`antclans`
B	Bernard	`bernard`
C	CEO Queue	`ceo`
D	Crazy Math	`crazy`
E	Fortunate Farmland	`farmland`
F	Mo' Money	`money`
G	Grand Theft Otto	`otto`
H	Safe Square	`square`
I	Trading Cards	`trading`
J	Clips of Video	`video`

Problem A: Ant Clans

Filename: `antclans`

Time limit: `5 seconds`

An ant dynasty has decided to move into a giant ant hill consisting of *n* burrows. The dynasty has a complicated social structure formed by a coalition of *k* ant clans, where each clan doesn't get along well with the others. To keep the peace, the ant emperor wants to partition the burrows into *k* equally-sized districts (one per clan), so that no ant from any clan should be able to reach any ant of an opposing clan. After deciding on the districts, the ants will drill tunnels between certain carefully chosen pairs of burrows so ants in each clan can travel among all the burrows allocated to the district for the clan. The ant emperor would like to know the cheapest cost possible for forming his districts and building the resulting tunnels.

Input

The first line of input contains 3 space separated integers: ***n*** (1 ≤ ***n*** ≤ 20), ***m*** (1 ≤ ***m*** ≤ $\frac{n(n-1)}{2}$) and ***k*** (1 ≤ ***k*** ≤ ***n***), representing the number of burrows, number of possible tunnels that can be drilled, and the number of districts to form. Burrows are labeled with identifiers [1, ***n***] .

This is followed by m lines each containing 3 space separated integers: **i** (1 ≤ ***i*** ≤ ***n***), ***j*** (1 ≤ ***j*** ≤ ***n***, ***i*** ≠ ***j***), and ***c*** (1 ≤ ***c*** ≤ 100) meaning that the burrow labeled i can be connected by a tunnel to the burrow labeled j with cost ***c***.

Output

On a line by itself, print the minimum cost possible of the desired partition, or -1 if such a district plan is impossible.

Samples

Input	**Output**

`4 4 2`
`1 2 300`
`2 3 200`
`3 4 100`
`4 1 8`
	`208`

`6 4 3`
`1 2 3`
`2 3 4`
`4 5 6`
`5 6 7`
	`-1`

Problem B: Bernard

Filename: `bernard`

Time limit: `2 seconds`

Bernard the bear has finally woken up from hibernation and is very hungry. He walks over to the nearest river and sees all the different kinds of fish swimming in a line, some fish more delicious looking than others. Sneaking up behind one of the fish, Bernard swipes and catches the fish. In the process, he was seen by the nearby fish and they swam away, not looking to become lunch today. Distraught that he can't catch all the fish for his lunch, Bernard now wonders what the most delicious lunch he can have given that some of the nearby fish will swim away after each time he catches one.

Given the deliciousness of fish in their current order in the river and an integer representing the distance of which fish can see, find the max sum of deliciousness that Bernard can have for his lunch. Each fish is located one unit away from another. If the fish can see k units away, and Bernard catches a fish at location x, the fish from [x – k, x – 1] and [x + 1, x + k] will swim away and cannot be caught.

Input

The first line will contain two space separated integers, $\boldsymbol{n}$ ($1 \le \boldsymbol{n} \le 10^5$) and $\boldsymbol{k}$ ($1 \le \boldsymbol{k} \le 10$), representing the number of fish in the river and the distance away the fish can see Bernard catch another fish. The next line will contain $\boldsymbol{n}$ space separated integers $\boldsymbol{x_i}$ ($1 \le \boldsymbol{x_i} \le 10^9$), representing the deliciousness of fish *i*.

Output

Output a single integer on a line by itself, representing the maximum sum of deliciousness that Bernard can have for his lunch.

Samples

Input	Output

`10 2`
`20 4 12 17 19 15 2 5 1 13`
	`52`

`10 4`
`8 28 1 9 28 2 9 1 27 2`
	`55`

Problem C: CEO Queue

Filename: `ceo`

Timelimit: `3 seconds`

The Grandestine made a two player online game similar to chess but with more pieces and free-to-play mechanics to rake in the bucks. Each player is given a rating that starts out at 0 and goes up or down as they win or lose games. Grand implemented a queue system that attempts to pair up players with similar ratings. Whenever a player enters the queue the game creates a range centered around that player's rating that expands over time which specifies acceptable opponent ratings. The range expands by 1 in each direction per unit time. For example, if a player rated 2050 entered the queue at time 0, at time 40 that player's range will be from 2010 to 2090. For a pair of players to be matched up, at least one of the players' ratings must fall within their opponent's rating range. Expanding on the previous example, if another player entered the queue at time 70 with rating 2150 he could be paired up with the first player at time 100 when the first player's range is 1900 to 2150 and the second player's range is 2120 to 2180. If instead the second player had a rating of 1990 he would have been immediately paired with the first player when he entered the queue at time 70 when the first player's range was 1980 to 2120. The queue pairs up players as soon as possible. In the event that multiple pairs of players can be matched at the same time, priority is given to the pair that contains the player who has been in the queue the longest. If there are multiple pairs that share this player, the tiebreaker is how long the other player of the pair has been in the queue. It is guaranteed that players do not join the queue at the same time and that no two players share the same rating.

Once a player is paired up for a match, the player will not be placed back into the queue. Thus, all players play either 0 or 1 match.

Grand is interested in the queue wait times for the players. In particular he wants to know how many players spent at least ***w*** time units waiting in the queue without being matched.

Input

The first line of input contains two space separated integers ***n*** ($1 \le n \le 10^5$) and ***w*** ($1 \le w \le 10^8$), the number of players that will join the queue and the number of minutes for the input query. The following ***n*** lines will contain information about each player. The i^th of these lines will contain two space separated integers: ***q_i*** ($0 \le q_i \le 10^8$) and ***r_i*** ($0 \le r_i \le 10^8$), the time the i^th player enters the queue and the i^th player's rating, respectively. The players are given in strictly increasing order of queue times.

Output

Output a single integer, the number of players that waited at least ***w*** time units in the queue without being matched.

Samples

Input	Output

`3 10`
`10 1000`
`11 1200`
`50 1001`
	`2`

`6 200`
`0 0`
`20 1000`
`30 510`
`40 400`
`50 300`
`320 500`
	`4`

Explanation of Sample Input 1:
The players rated 1000 and 1001 get matched up at time 50. The player rated 1200 is stuck in the queue forever.

Explanation of Sample Input 2:
-The players rated 510 and 400 get paired at time 140
-The players rated 0 and 300 get paired at time 300
-The players rated 1000 and 500 get paired at time 520

Problem D: Crazy Math

Filename: `crazy`
Time limit: `3 seconds`

Arup loves math. Many love the Fibonacci sequence, but true mathematicians love to generalize results. As such, Arup prefers looking at the Generalized Fibonacci Sequence that is defined as follows, where a and b are constants:

g(0) = a
g(1) = b
g(n) = g(n-1) + g(n-2), for all integers n > 1

For various Generalized Fibonacci Sequences, Arup would like to know the nth term of the sequence. Can you write a program to do it for him? Additionally, since Generalized Fibonacci numbers get very big quickly, Arup would like you to calculate the final result MOD 10^9.

Input

The input consists of three non-negative, space separated integers: ***a*** (***a*** ≤ 100), ***b*** (***b*** ≤ 100), and ***n*** (***n*** ≤ 2^{48}), the first term of the Generalized Fibonacci sequence, the second term of the Generalized Fibonacci sequence and the Generalized Fibonacci number Arup wants you to calculate, respectively.

Output

The output will be a single integer, the nth Generalized Fibonacci number MOD 10^9, corresponding to the input.

Samples

Input	**Output**
`3 4 1`	`4`

Input	**Output**
`17 6 2`	`23`

Input	**Output**
`1 1 3749999997`	`499999999`

Problem E: Fortunate Farmland

Filename: `farmland`
Time limit: `2 seconds`

The country of Fortunate Farmland judges its success differently than most nations. Rather than wanting to achieve a maximum GDP or maximize the number of millionaires in the country, the country aims to maximize the minimum income of its citizens. In order to improve this minimum value, the country has a fund set aside (collected from taxes) and will use all the money in that fund to raise incomes in such a way as to maximize the minimum income of all citizens.

Take the following small example. Let's say that there are 10 citizens with the following initial incomes in Farmland Dollars: 12, 8, 3, 9, 15, 22, 27, 13, 77 and 18. Now, let's say that the Fortunate Farmland government has a total of 30 Farmland Dollars to distribute amongst its citizens. Note that we can only pay integer amounts of Farmland Dollars to each citizen. The government would pay 3 Farmland Dollars to the first person, 7 to the second person, 12 to the third person, 6 to the fourth person and 2 to the 8th person. After these payments, the new income list (in Farmland Dollars) would be: 15, 15, 15, 15, 15, 22, 27, 15, 77 and 18. Now, the minimum income of any citizen is 15 Farmland Dollars.

Given the initial incomes of each citizen in Fortunate Farmland as well as how much money the government of Fortunate Farmland has to distribute to each citizen (in integer increments of Farmland Dollars) determine the minimum income of any citizen of Fortunate Farmland after the money has been distributed. As previously stated, the government of Fortunate Farmland always distributes this money in such a way as to maximize the minimum income of its citizens, within the restrictions of giving integer Farmland Dollars to some of its citizens.

Input

The first line of input will contain two space-separated positive integers, $\boldsymbol{c}$ ($\boldsymbol{c} \le 10^5$), representing the number of citizens of Fortunate Farmland and $\boldsymbol{t}$ ($\boldsymbol{t} \le 10^{12}$, $\boldsymbol{t/c} \le 10^9$), the amount Farmland Dollars to be distributed amongst the citizens. The following $\boldsymbol{c}$ lines contain one positive integer each, $\boldsymbol{m_i}$ ($\boldsymbol{m_i} \le 10^9$), representing the income of the ith citizen of Fortunate Farmland in Farmland Dollars.

Output

Output a single line with the minimum income amongst all the citizens of Fortunate Farm after the government makes its distribution.

Samples

Input	Output

10 30
12
8
3
9
15
22
27
13
77
18
	15

5 100
1
2
3
4
10
	24

Problem F: Mo' Money

Filename: `money`

Time limit: `2 seconds`

Max has a number of coins and wants to know how many ways he can use some or all of the coins to reach a target value. Can you write a program to help him out?

Input

The first line of input will contain two space separated positive integers, $\boldsymbol{n}$ ($\boldsymbol{n} \le 15$), the number of coins Max has, and $\boldsymbol{t}$ ($\boldsymbol{t} \le 10^6$), the target value he wants to reach. The second line of input will contain n space separated positive integers, $\boldsymbol{a_1}$ through $\boldsymbol{a_n}$, where $\boldsymbol{a_i}$ ($\boldsymbol{a_i} \le 5*10^5$) is the value of the ith coin.

Output

Output a single integer on a line by itself, representing the number of ways Max can combine his coins to reach his target value.

Samples

Input	**Output**

`6 20`
`3 10 4 7 3 6`
	`6`

`10 3500`
`1000 500 750 250 100 800 1200 900 1300 3000`
	`9`

Explanation of Sample Input 1:

Here are the sets of coins that add up to a value of 20:

1. {coin1, coin2, coin3, coin5}
2. {coin1, coin2, coin4}
3. {coin1, coin3, coin4, coin6}
4. {coin2, coin3, coin6}
5. {coin2, coin4, coin5}
6. {coin3, coin4, coin5, coin6}

Notice that this set happens to have two different coins with the same value, 3, but the two coins are treated as distinct coins.

Problem G: Grand Theft Otto

Filename: `otto`
Time limit: `1 second`

Otto is late for the programming contest that he's supposed to compete in today! The time now is t = 0; Otto just woke up at his house, which is $\boldsymbol{d_1}$ meters away from the site of the contest. Luckily, Otto prepared so much for waking up yesterday that he can immediately leave his house and jump into his car, which is also considered $\boldsymbol{d_1}$ meters away from the contest at t = 0.

Because Otto is also a certain type of superhero (or supervillain, depending on perspective), he can transform his body into electricity and possess machines (like cars) and start driving them immediately. That's actually how he starts using his car as he leaves his house.

The road he takes from his house to the contest site is a straight-line road with $\boldsymbol{n}$ cars parked on the side of the road, including Otto's car. Each of these cars is indexed 1 through $\boldsymbol{n}$, and car *i* is parked $\boldsymbol{d_i}$ meters away from the contest site. While possessing car indexed *i*, Otto can travel $\boldsymbol{v_i}$ meters per second towards the contest site, without needing any time to accelerate. He can also instantaneously switch the car that he is possessing with another whenever the other car is parked and the same distance from the contest as the car he is currently possessing. In such a situation, the new car will instantaneously accelerate to its fastest speed and the old car will stop immediately. Of course, such a switch of car usage does not always have to happen.

Input

The first line of input contains a single positive integer $\boldsymbol{n}$ ($\boldsymbol{n} \leq 10^5$), the number of cars. $\boldsymbol{n}$ lines follow, each with two space-separated positive integers each. The *i*th of these lines will denote that car *i* is parked $\boldsymbol{d_i}$ meters away from the contest site and has a top speed of $\boldsymbol{v_i}$ meters per second. The cars will be listed in decreasing order of their distance from the contest.

Output

Output the minimum amount of time it will take for Otto to arrive at the contest site, in seconds. Solutions will be considered correct if their output differs from the judge output by less than 10^{-4} seconds.

Samples

Input	Output

2
4 1
2 2
	3.00000

3
5 1
4 2
3 3
	2.50000

5
7 2
5 6
4 3
4 6
3 9
	1.66667

Problem H: Safe Square

Filename: `square`

Time limit: `1 second`

Simon lives in a rectangular grid. At any given time, he occupies a single square on the grid. Some of the squares on the grid he lives in have snakes and he can not occupy those squares. In fact, he dislikes snakes so much, that he prefers not to occupy any square where half or more of the adjacent squares contain snakes. Two squares are adjacent if they share a corner, so the maximum number of squares adjacent to any square is 8. Simon considers these squares to be dangerous squares, as well as squares with snakes. He considers all other squares to be safe squares. Help Simon figure out how many squares on the grid in which he lives are safe squares.

Input

The first line of input contains two space separated integers, ***r*** ($1 \le r \le 100$) and ***c*** ($1 \le c \le 100$, $rc > 1$), the number of rows and columns, respectively, on the grid in which Simon lives. The following ***r*** lines contain ***c*** characters each, either '.' or 'S'. A dot (.) indicates a square without a snake and a capital S indicates a square with a snake. The j^th character on the i^th of these lines represents the contents of the square on row i, column j of the grid in which Simon lives.

Output

Output a single integer representing the number of safe squares on the input grid in which Simon could live.

Samples

Input	Output

`3 5`
`.....`
`S.S.S`
`SS.S.`
	`6`

`5 6`
`S.S.S.`
`.S.S.S`
`S.S.S.`
`.S.S.S`
`S.S...`
	`4`

Problem I: Trading Cards

Filename: `trading`
Time limit: `1 second`

Alice and Carl love collecting trading cards. They both have large collections, and want to know how much their collections are worth. Both of their collections consist of several types of cards, each with an associated value.

Input

The first line of input consists of a single integer, $\boldsymbol{n}$ ($1 \le \boldsymbol{n} \le 200$), the number of card types Alice has. The second line of input consists of $\boldsymbol{n}$ integers, $\boldsymbol{a_1}$ through $\boldsymbol{a_n}$, where $\boldsymbol{a_i}$ represents the value of Alice's ith card type ($1 \le \boldsymbol{a_i} \le 100$). The third of input consists of $\boldsymbol{n}$ integers, $\boldsymbol{b_1}$ through $\boldsymbol{b_n}$, where $\boldsymbol{b_i}$ represents how many cards of the ith type Alice has ($1 \le \boldsymbol{b_i} \le 100$).

The fourth line of input consists of a single integer, $\boldsymbol{m}$ ($1 \le \boldsymbol{m} \le 200$), the number of card types Carl has. The fifth line of input consists of $\boldsymbol{m}$ integers, $\boldsymbol{c_1}$ through $\boldsymbol{c_m}$, where $\boldsymbol{c_i}$ represents the value of Carl's ith card type ($1 \le \boldsymbol{c_i} \le 100$). The sixth of input consists of $\boldsymbol{m}$ integers, $\boldsymbol{d_1}$ through $\boldsymbol{d_m}$, where $\boldsymbol{d_i}$ represents how many cards of the ith type Carl has ($1 \le \boldsymbol{d_i} \le 100$).

Output

On a single line, output two space separated integers: the value of Alice's collection and the value of Carl's collection, respectively.

Samples

Input	Output

`4`
`1 2 3 4`
`3 5 3 1`
`3`
`10 4 60`
`3 20 1`
	`26 170`

Problem J: Clips Of Video

Filename: `video`
Timelimit: `5 seconds`

A Youtube Poop is a type of video which is a modification of another video, in which the modified video consists of several (possibly duplicate) clips of someone speaking in the original video, with an ordering which makes it sound like the speaker is saying something else. Youtube Poops are usually considered humorous and crude, but are almost universally considered of much higher quality when the Youtube Poop does not need to use as many clips to produce the desired mangled-up text the creator of the Youtube Poop intends.

Let's say string ***A*** represents the speech given in an original video, and we want to create a Youtube Poop of the original video in which string ***B*** represents the speech given in our Youtube Poop. What, then, is the minimum amount of substrings of ***A*** needed to form string ***B***?

Input

The first line of input consists of the string ***A*** ($1 \leq |A| \leq 10^5$). The second line of input consists of the string ***B*** ($1 \leq |B| \leq 10^5$). All strings will only contain lower-case Latin letters.

Output

Output the minimum number of substrings of ***A*** necessary to form ***B***. If it is not possible to take substrings of ***A*** to make ***B***, output -1.

Samples

Input	Output
`abcbcd` `abcd`	`2`
`iamsmart` `iamdumb`	`-1`
`asmallmallinmalta` `atallmallinlima`	`5`

Explanation of Sample Input 3:
The first substring is "a" starting at position 17 in ***A***, followed by "ta" starting at position 16, followed by "llmallin" starting at position 5, followed by "li" starting at position 10, followed by "ma" starting at position 3.