Greedy Santa (`christmas`)

Santa Claus loves everything about his job: the sled, the reindeers, the chilly weather, the smiles of children when they receive their gifts... everything except one thing: he really hates his managers.

As a matter of fact, the *North Pole Holdings LLC* is a very demanding company, and its top executives are arrogant and hard to please. Every year Santa tries his best to spend as little as possible to buy the gifts, be as fast as possible to deliver them in time, and yet there is always some manager complaining!

That is why, this year, Santa feels a little "vindictive": he will buy very expensive gifts (for example, he's thinking about buying a vacation to Hawaii as a gift for himself) in order to squander **all of the budget** provided by the company, a total of B euros.

Figure 1: Santa Claus, enjoying his vacation paid for by the company.

Santa wants to use all of the company's budget **even if that means having to pay something out of his own pockets!** In other words, he wants to choose a subset of the available gifts that *maximizes* money spent from the budget while *minimizing* the amount of money he has to spend out of his pockets.

Help Santa by computing the total cost of the gifts!

☞ Among the attachments of this task you may find a template file `christmas.*` with a sample incomplete implementation.

Input

The first line contains two integers N and B, respectively the number of available gifts and the budget set up by Santa's managers. The second line contains N integers G_i, where the i-th integer represents the cost of the i-th gift.

Output

You need to write a single line with an integer: the total cost of some subset of the gifts such that the budget is used as much as possible while minimizing what Santa will have to pay by himself.

Constraints

- $1 \leq N \leq 100$.
- $1 \leq B \leq 4000$.
- $1 \leq G_i \leq 100$ for each $i = 0 \ldots N-1$.
- The budget is always sufficient to buy at least one of the gifts.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (30 points)	The cost of all the gifts is either 1 or 2.
– **Subtask 3** (50 points)	$N \leq 20$.
– **Subtask 4** (20 points)	No additional limitations.

Examples

input	output
`5 30`	`34`
`13 7 21 8 14`	
`10 10`	`10`
`1 2 1 1 2 1 1 1 2 1`	
`5 100`	`60`
`10 11 12 13 14`	

Explanation

In the **first sample case**, Santa could choose to buy the gifts with prices 7, 8 and 14 for a total of 29 euros, but that would not use all of the budget. The cheapest way to use as much of the budget as possible is to buy gifts 13 and 21, for a total of 34 euros. Santa will need to pay 4 euros out of his pockets.

In the **second sample case**, it might seem like a good solution to buy all the 1s (thus reaching 7) then buy a 2 (reaching 9) and finishing off with another 2, reaching 11 euros. However, it's smarter to buy all of the 2s first (thus reaching 6) and then buy the 1s until reaching exactly 10 euros.

In the **third sample case** there is sadly no way to use all of the budget.

Pet Fair (exhibitions)

Edoardo and Luca decided to diversify their businesses by organizing an astounding *pet fair*. However, they are already fighting for its organization, since Luca is a huge fan of *dogs* and Edoardo a strong supporter of *cats*!

This fair consists in N different exhibitions, each of them with a specific *awesomeness coefficient* A_i (positive integer), involving a certain *pet* P_i (a dog or a cat), and presented by a certain *guide* G_i (a dog-lover or a cat-lover), for $i = 0 \dots N-1$. Note that a cat exhibition is not necessarily presented by a cat-lover, and similarly for dogs.

Figure 1: Long-standing enemies confronting for the ultimate supremacy.

During the fair, M groups of tourists will arrive, starting from different exhibitions E_j for $j = 0 \dots M-1$. The guides, after presenting their exhibition to the incoming group, will direct the group to another exhibition of their choice. The groups will continue to follow the instructions of the guides, until they will reach in this way an exhibition they have already seen: at that point, the group will leave the fair.

Edoardo and Luca know very well that, when a group leaves, only the most awesome exhibition encountered during the visit will be remembered... and they want this memory to be of their beloved pet! Edoardo and Luca are very smart, so they have developed the best possible strategies for their sides and have coordinated tactics with their fellow guides accordingly. Calculate which pet will be remembered by each incoming group, knowing that every guide will follow the best possible strategy for their side!

☞ Among the attachments of this task you may find a template file `exhibitions.*` with a sample incomplete implementation.

Input

The first line contains integers N and M. The second line contains M integers E_i. The following N lines contain three integers A_i, P_i, G_i each.

Output

You need to write a single line with M integers R_i, corresponding to the pet that will be remembered by each group, assuming a perfect strategy from both competing sides.

Constraints

- $2 \leq N, M \leq 100\,000$.
- $0 \leq A_i \leq 10^9$ for each $i = 0 \dots N-1$.
- $0 \leq E_i \leq N-1$ for each $i = 0 \dots M-1$.
- P_i, G_i, R_i can be 0 to indicate *dogs*, or 1 to indicate *cats*.
- The values A_i are all distinct.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (10 points)	$N = 2$.
– **Subtask 3** (10 points)	$N = 3$.
– **Subtask 4** (15 points)	$N, M \leq 10$.
– **Subtask 5** (25 points)	$N, M \leq 100$.
– **Subtask 6** (20 points)	$P_i = G_i$ for all i.
– **Subtask 7** (20 points)	No additional limitations.

Examples

`input.txt`	`output.txt`

`2 3`
`1 0 1`
`50 1 1`
`80 0 1`
	`0 0 0`

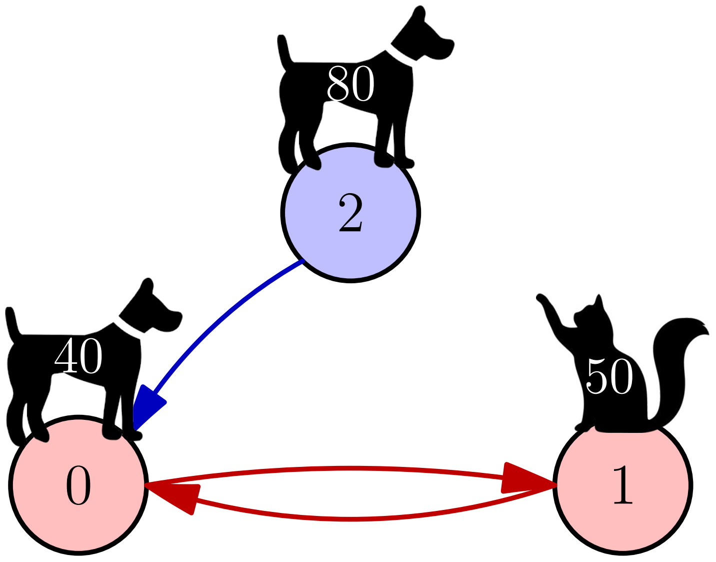
`3 3`
`0 1 2`
`40 0 1`
`50 1 1`
`80 0 0`
	`1 1 0`

`5 3`
`4 0 2`
`30 0 0`
`10 0 0`
`20 1 1`
`90 1 0`
`70 0 1`
	`1 0 1`

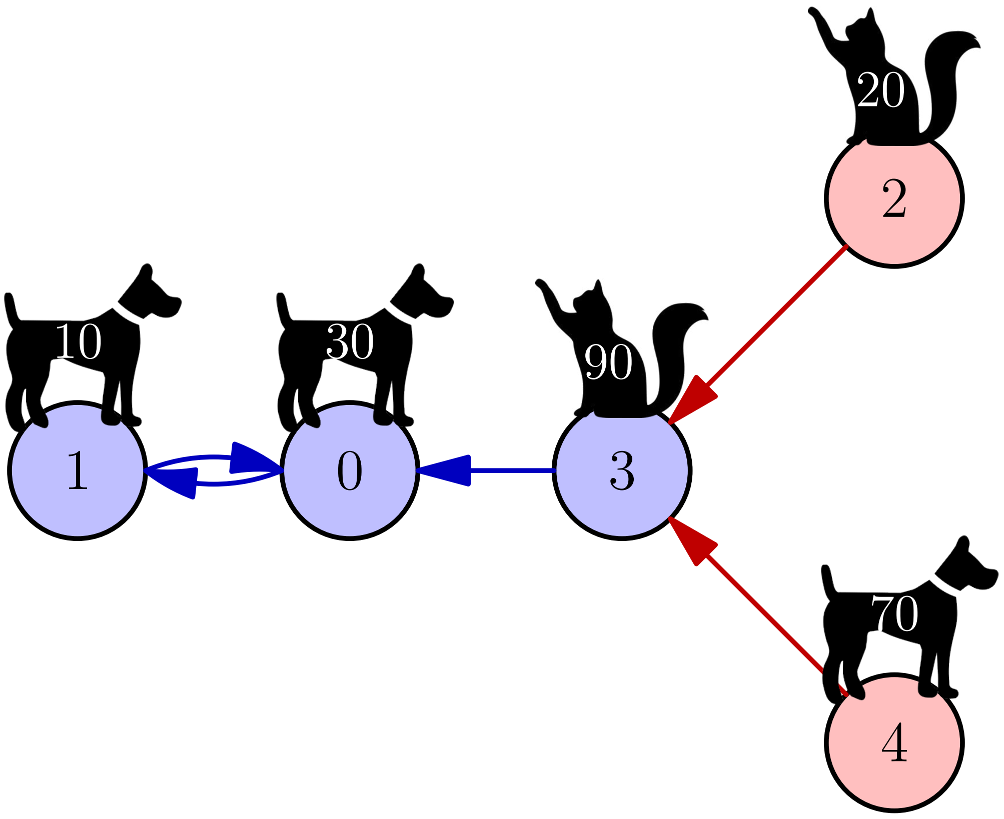
Explanation

In the **first sample case**, every exhibition is controlled by a cat lover. However, both exhibitions cannot do anything but directing groups to the other exhibition. Thus, every group will see every exhibition and leave with the memory of the best among them (which is exhibition 1 of dogs).

In the **second sample case**, an optimal strategy is represented by the arrows in the following picture, where blue exhibitions are controlled by dog-lovers and red exhibitions are controlled by cat-lovers, and awesomeness is written in white on top of the pet. According to this strategy, dogs are remembered only if starting from exhibition 2.



In the **third sample case**, an optimal strategy is the following: according to this strategy, dogs are remembered only if starting from exhibitions 0 or 1.



Master Chef (kitchen)

William is attending the finals of Master Chef Italia! For this last round of selection he will have to prepare N different dishes, each of them taking H_i units of heat in order to be cooked. Time is very limited, so he needs to plan carefully in order to cook everything in time. For this round, William will be provided of M stoves of various sizes, each of them taking T_i seconds to emit a full unit of heat (larger stoves are faster, smaller ones are slower).

On each stove, William can put only **one** of the dishes he has to prepare at a time: he may put other ones only **after** the dishes previously cooked there are ready. Furthermore, each dish can only be put on a **single** stove from the beginning until the end: changing stoves halfway through cooking may seriously impair the quality of the prepared dishes.

Figure 1: Some stoves at the location of the Master Chef Italia finals.

After spending half of his time planning, William is still puzzled about how to schedule the different dishes on the various stoves. Help William by telling him on which cooker to put every dish, in order to finish as soon as possibile!

☞ Among the attachments of this task you may find a template file `kitchen.*` with a sample incomplete implementation.

Input

The first line contains two integers N and M. The second line contains N integers H_i. The third line contains M integers T_i.

Output

You need to write two lines. The first line contains the total time you achieved for cooking everything. The second line contains N integers: the stoves (indexed from 0 to $M-1$) where to put each dish.

Constraints

- $1 \le N, M \le 10\,000$.
- $1 \le H_i \le 100$ for each $i = 0 \ldots N-1$.
- $1 \le T_i \le 100$ for each $i = 0 \ldots M-1$.
- The time to put and take a dish from a stove is zero.

Scoring

Your program will be tested against 20 different test cases. The score of your solution will be equal to the sum of the scores for all the test cases.

Your program will score zero points in a test case if the total time provided does not agree with the provided positions, or if the output format is invalid. Otherwise, the score is computed as follows:

$$\text{score} = 5 \cdot \min\left(\frac{T_{\text{official}}}{T_{\text{obtained}}}, 1\right)$$

where T_{official} is the time achieved by the official solution, and T_{obtained} is the time achieved by your program. Notice that the official solution is **not** granted to be optimal!

The contest platform will not show subtasks for this task explicitly, however, the test cases are designed in order to reflect the following subtasks:

– **Subtask 1** (10 points) $M = 1$ and $T_0 = 1$.

– **Subtask 2** (10 points) $M = 1$.

– **Subtask 3** (20 points) $T_i = 1$ for each i.

– **Subtask 4** (20 points) $N \le M$.

– **Subtask 5** (10 points) $N, M \le 5$.

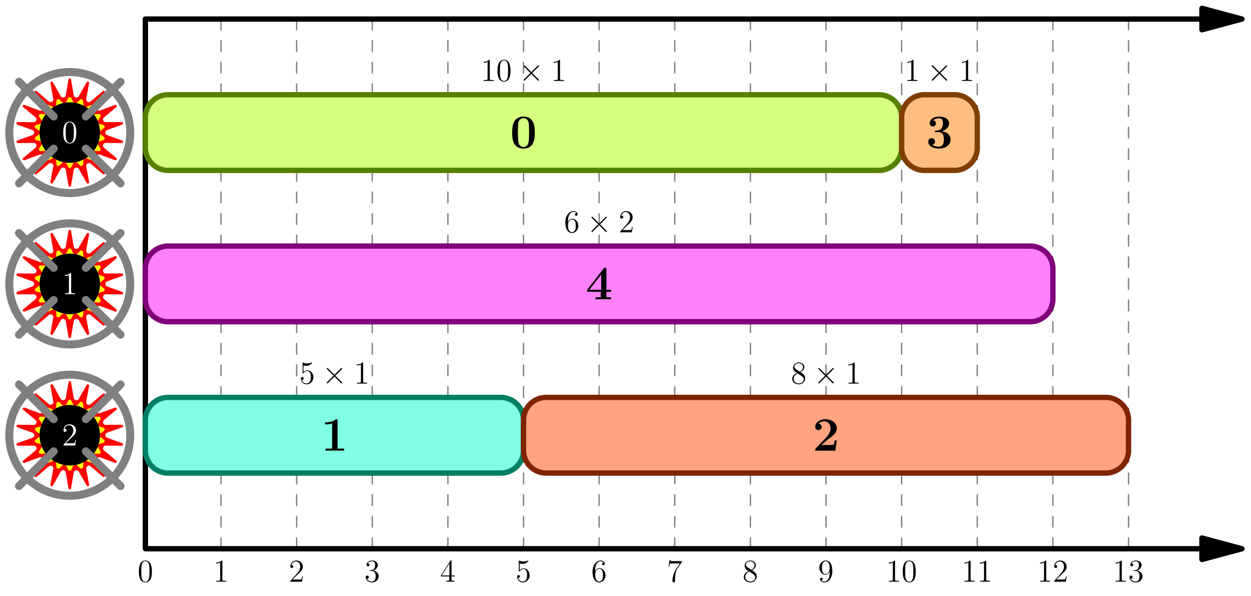
– **Subtask 6** (30 points) No additional limitations.

Examples

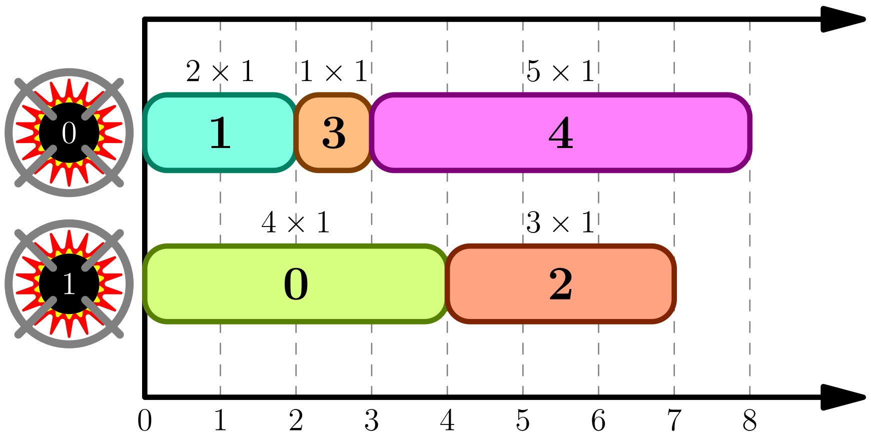
input	output
`5 3`	`13`
`10 5 8 1 6`	`0 2 2 0 1`
`1 2 1`	
`5 2`	`8`
`4 2 3 1 5`	`1 0 1 0 0`
`1 1`	

Explanation

In the **first sample case**, a possible plan is the following.



In the **second sample case**, a possible plan is the following:



Shopping Malls (`malls`)

Luca's friend is tired of all new technological startup being created and, in response, wants to start a traditional business: a shopping mall.

Before constructing the physical structure, she plans to carefully analyze the locations of already existing malls, as she obviously does not want to see her business cannibalized by competitors.

Figure 1: One of the many shopping malls near Bergamo.

On the path from Milan to Bergamo (near where Luca lives) there is an astonishing number of big malls, approximately aligned on a "virtual straight road". With the help of some acquaintances, she has obtained information about N malls. For each one, it is known the distance in kilometers from the center of Milan, considered at distance 0 by convention.

The new mall must be constructed within this "virtual road" of lenght K, extremes included. Luca has been called to find the best position for his friend's new activity: where should he advise her to build the mall, in order to *maximize* the *minimum* distance with an existing mall?

☞ Among the attachments of this task you may find a template file `malls.*` with a sample incomplete implementation.

Input

The first line contains two integers N and K, respectively the number of malls and the length (in kilometers) of the "virtual road" they are approximately placed on. The second line contains N integers D_i, the distances (in kilometers) of the malls.

Output

You need to write a single line with an integer: the optimal distance at which the new mall should be constructed.

Constraints

- $1 \leq N \leq 100\,000$.
- $10 \leq K \leq 1\,000\,000\,000$.
- $N < K$.
- $0 \leq D_i \leq K$ for each $i = 0 \ldots N-1$.
- There are never two shopping malls at the same position.
- If there are multiple optimal solutions, you can output any of them.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (10 points)	$N = 1$.
– **Subtask 3** (40 points)	$N \leq 100$, $K \leq 1\,000$.
– **Subtask 4** (25 points)	$N \leq 500$
– **Subtask 5** (25 points)	No additional limitations.

Examples

input	output

`5 50`
`17 4 36 41 44`
	`27`

`3 50`
`22 34 41`
	`0`

Explanation

In the **first sample case** an optimal solution is to construct the new mall at distance 27. The nearest competitor is 9 kilometers away, at distance 36.

In the **second sample case** the smartest move is to construct the new mall right in the center of Milan (at distance 0, by definition). The nearest competitor is 22 kilometers away, at distance 22.

Muffin Selection (`muffin`)

You may not know that William is an expert about muffins: he is even able to immediately judge the taste T_i of a muffin without even biting it! He is now in front of a long row of N muffins, and he really wants to eat some of them.

Figure 1: Some muffins aligned in a row, ready to be tasted.

The only way to pick up a muffin is to use a wide paddle that can lift exactly K muffins at a time, precisely the number that William wants.

Help him choosing the best range of K adjacent muffins with the highest sum of tastes, to maximize his satisfaction!

☞ Among the attachments of this task you may find a template file `muffin.*` with a sample incomplete implementation.

Input

The first line contains two integers N and K, respectively. The second line contains N integers T_i.

Output

You need to write a single line with an integer: the maximum possible sum of tastes.

Constraints

- $1 \le N \le 1\,000\,000$.
- $1 \le K \le N$.
- $-1000 \le T_i \le 1000$ for each $i = 0 \ldots N-1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)	Examples.
– **Subtask 2** (10 points)	$T_i > 0$ for each $i = 0 \ldots N-1$.
– **Subtask 3** (10 points)	$K = 1$.
– **Subtask 4** (20 points)	$K = 2$.
– **Subtask 5** (30 points)	$N \le 1000$.
– **Subtask 6** (30 points)	No additional limitations.

Examples

`input.txt`	`output.txt`
`7 3` `10 -3 -1 6 4 1 -10`	`11`
`5 2` `1 -2 4 -8 16`	`8`

Explanation

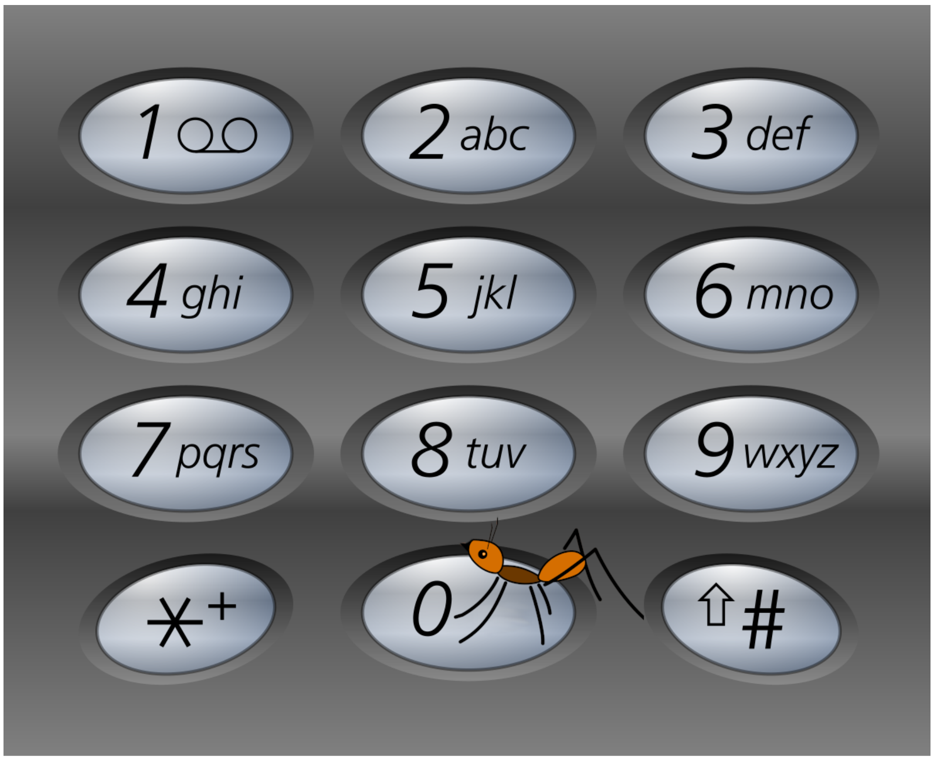
In the **first sample case** the best choice is to take the muffins having taste `6 4 1` with a total taste of $6 + 4 + 1 = 11$.

In the **second sample case**, as you are forced to take exactly two muffins, the best choice is to take the last two ones, with a total taste of $-8 + 16 = 8$.

Call for Help (numpad)

Edoardo was playing with a new device when suddenly he pushed the wrong button and he was transformed into an ant! In order to try and reverse the effect of the device, Edoardo is now trying to call the helpline of the company that made it.

The toll-free phone number is written on the back of the device, and is formed by N digits. Edoardo is initially located on top of his phone numpad, specifically on the number "0".



At any moment, Edoardo can choose to either:

- Jump on the spot. This will *press* the number where he's located.
- Jump in the direction of some adjacent number: this will change his location but will *not press* the number after "landing". He can move only in 4 directions: up, down, left, right.

Compute how many jumps he will need to perform to dial the whole number.

☞ Among the attachments of this task you may find a template file `numpad.*` with a sample incomplete implementation.

Input

The first and only line contains a string S formed by N characters. The string represents the phone number Edoardo needs to call.

Output

You need to write a single line with an integer: the number of jumps Edoardo needs to perform to dial the whole number.

Constraints

- $1 \leq N \leq 1\,000\,000$.
- Characters of S are always numeric digits (from `0` to `9`).

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (30 points) The helpline number is always formed by only two digits: 0 and 8.

– **Subtask 3** (20 points) $N \leq 10$.

– **Subtask 4** (20 points) $N \leq 10\,000$.

– **Subtask 5** (30 points) No additional limitations.

Examples

`input`	`output`
`088`	`4`
`01`	`6`
`010`	`11`

Explanation

In the **first sample case** Edoardo can first jump on the spot, then jump from 0 to 8, then jump on the spot, and finally jump again on the spot.

In the **second sample case** Edoardo can jump on the spot, then jump from 0 to 8, then from 8 to 5, then from 5 to 4, then from 4 to 1, then jump on the spot.

Tournament Planning (poker)

Giorgio is so confident in his poker skills that he is planning his participation in the next N tournaments *based on the assumption that he will always win.*[1] However, the plan is not trivial since some tournaments may overlap, and others may be too expensive to participate in!

More precisely, each tournament i for $= 0 \ldots N-1$ is held in day D_i, starts at time S_i, ends at time E_i, has a buy-in[2] B_i and a prize P_i. Before the start of the tournaments, Giorgio was able to borrow M euros from friends to spend on buy-ins.

Now, Giorgio has to select a subset of the tournaments such that for each i among them:

Figure 1: Giorgio practicing his infallible poker skills.

- every other selected tournament j on the same day does not overlap (i.e., either $S_i \geq E_j$ or $S_j \geq E_i$);
- Giorgio is able to pay for the buy-in B_i, assuming to have collected all the prizes for the selected tournament j finished before the start of i.

Of course, the selection should allow Giorgio to earn the most euros possible. Help Giorgio plan his next holidays by computing the amount of euros he will have (at most) after all the tournaments!

☞ Among the attachments of this task you may find a template file `poker.*` with a sample incomplete implementation.

Input

The first line contains the integers N, M: the number of tournaments and the "starting" money. Each of the following N lines contains five integers D_i, S_i, E_i, B_i, P_i: a description of the i-th tournament.

Output

You need to write a single line with an integer: the maximum amount of euros that Giorgio can have after all the tournaments.

Constraints

- $1 \leq N, M \leq 100\,000$.
- $0 \leq B_i \leq P_i \leq 10^9$ for each $i = 0 \ldots N-1$.
- $0 \leq D_i \leq D_{i+1} \leq 1000$ for each $i = 0 \ldots N-2$.
- $0 \leq S_i < E_i \leq 1000$ for each $i = 0 \ldots N-1$.

[1] This confidence has absolutely nothing to do with secret cheating tactics.
[2] The *buy-in* of a poker tournament is the participation fee required to access it.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
- **Subtask 2** (10 points) There is at most one tournament every day.
- **Subtask 3** (15 points) $S_i = 0$ and $E_i = 1$ for each i.
- **Subtask 4** (25 points) $B_i = 0$ and $P_i = 1$ for each i.
- **Subtask 5** (20 points) $N \leq 10$.
- **Subtask 6** (15 points) $N \leq 1000$.
- **Subtask 7** (15 points) No additional limitations.

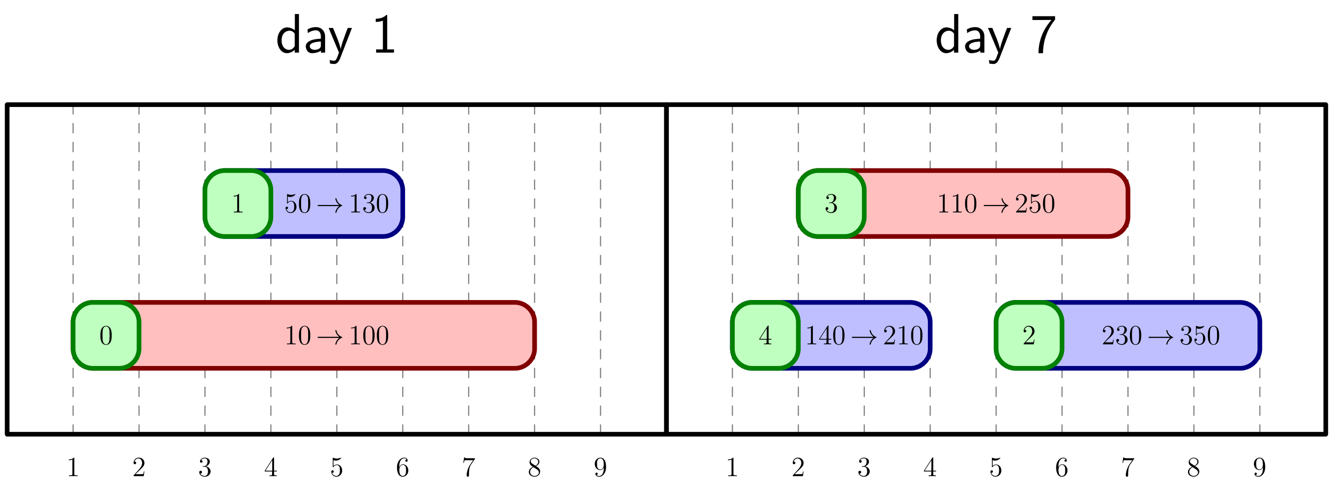
Examples

input	output
`3 100`	`600`
`2 0 10 100 300`	
`3 1 8  500 1000`	
`5 0 12 300 600`	
`5 50`	`280`
`1 1 8 10  100`	
`1 3 6 50  130`	
`7 5 9 230 350`	
`7 2 7 110 250`	
`7 1 4 140 210`	

Explanation

In the **first sample case**, the tournaments are all non-overlapping. By participating in the first tournament, Giorgio can have 300 euros by the end of day 2. Unfortunately, this is not enough to participate in the second tournament; it is however enough for the third tournament, at the end of which Giorgio will have 600 euros.

In the **second sample case**, the tournaments are arranged as in the following schedule.



In day 1, the best option is choosing tournament 0, at the end of which Giorgio will have $50-10+100 = 140$ euros. Even though tournaments 4 and 2 are non-overlapping in day 7, Giorgio cannot do both of them, since after tournament 4 he will not have enough money for the buy-in of tournament 2. Thus, he can just decide between tournament 4 and 3, among which the most convenient is number 3 for a final amount of $140 - 110 + 250 = 280$ euros.

N-Restaurants (`restaurants`)

Cooking reality shows are now widespread on television. Not being much different from everyone else, Giorgio occasionaly watches his favourite one: "quattro ristoranti" (Italian for "four restaurants"). The format is straightforward: alternately, all but one restaurateurs plus Alessandro Borghese (the host of the show) dine in the restaurant of the remaining restaurateur, giving each a mark from 0 to 10. The episode is won by the restaurateur who got the maximum score, defined as the sum of the marks of other restaurateurs plus the mark given by Alessandro Borghese.

Figure 1: Alessandro Borghese giving full score to a restaurateur ("diesci", in Italian).

Giorgio keeps receiving notifications from his smartphone and thus is not able to pay full attention to the television. Sometimes, right in the middle of the show, he wonders: who is winning at the moment? Help him by determining, after each round, who leads the ranking (or, in other words, who has the maximum score).

☞ Among the attachments of this task you may find a template file `restaurants.*` with a sample incomplete implementation.

Input

The first line contains the only integer N, the number of restaurateurs. There are N lines following, each containing $N+1$ integers. On the i-th of these lines, the first N integers represent the votes from other restaurateurs to restaurant i, while the $N+1$-th integer is the vote given by Alessandro Borghese. As a restaurateur cannot vote for themselves, the i-th value of the i-th line will always be -1, as a placeholder.

Output

You need to write N lines containing an integer each. The i-th line must contain the identifier (from 1 to N) of the restaurant that leads the ranking after the first i rounds.

Constraints

- $2 \leq N \leq 100$.
- Each (valid) vote is in the range 0-10.
- A restaurateur cannot vote for himself: the i-th vote of the i-th round is fixed at -1 as a placeholder and is irrelevant.
- It is guaranteed that after each round the leading restaurateur can be uniquely identified: there is never a tie.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (20 points) $N = 2$.

– **Subtask 3** (20 points) $N = 4$, as in the original TV show!

– **Subtask 4** (60 points) No additional limitations.

Examples

input	output

`2`
`-1 6 8`
`5 -1 6`
	`1`
	`1`

`3`
`-1 8 5 5`
`10 -1 8 9`
`8 9 -1 9`
	`1`
	`2`
	`2`

Explanation

In the **first sample case** there are only two restaurateurs. After the first round, the first restaurateur gets 6 points from the other one and 8 points from Alessandro Borghese, for a total of 14 points. After the second round, the second restaurateur gets 5 points from the first one and 6 points from Alessandro Borghese, for a total of 11 points. Thus, after both rounds, the first restaurateur has the best score.

In the **second sample case** there are three restaurateurs. After the first round, the first restaurateur gets 13 points from the other two and 5 points from Alessandro Borghese, for a total of 18 points. He thus trivially leads the ranking.

After the second round, the second restaurateur gets 18 points from the other two and 9 points from Alessandro Borghese, for a total of 27 points. In the current situation, the second restaurateur is the one with the maximum score.

After the third round, the third restaurateur gets 17 points from the other two and 9 points from Alessandro Borghese, for a total of 26 points. Thus, he conquers the second place, leaving the second restaurateur at the top of the ranking.