Saturday Night Game (collision)

Edoardo and Luca always try to come up with fun games that involve mathematics to entertain themselves on the Saturday nights in this pandemic.

They both have N empty slots to fill in with numbers. Before starting to play, the two friends agree on a special value K. The final goal of the game is to obtain two sequences with the same sum (modulo $10^9 + 7$).

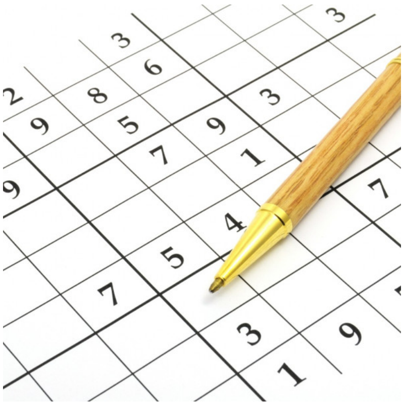


Figure 1: Couldn't Edoardo and Luca just play Sudoku? Well, those are kids' games!

The game is not easily over, as there are two very strict requirements:

- Numbers cannot be repeated within the same sequence **and** cannot be present in both sequences;
- The XOR (exclusive or) of the numbers in each sequence must be exactly K.

Finding those sequences, however, comes out to be harder than expected. Can you help them?

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

Input

The first line contains the two integers, N and K.

Output

You need to write a two lines with N integers each: two sequences for Luca and Edoardo that fulfill the game's requirements.

✍ The *modulo* operation ($a \bmod m$) can be written in C/C++ as `(a % m)` and in Pascal as `(a mod m)`. To avoid the *integer overflow* error, remember to reduce all partial results through the modulus, and not just the final result!

The *XOR* operation between a and b ($a \oplus b$) can be written in C/C++ as `(a ^ b)` and in Pascal as `(a xor b)`.

Constraints

- $2 \leq N \leq 100$.
- $1 \leq K \leq 10^9$.
- All non-negative integer values (from 0 to $2^{31} - 1$) are valid in the output sequences.
- Multiple correct solutions may exist: 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** (7 points) $N = 2, K$ is odd.

– **Subtask 3** (8 points) $N = 2, K \leq 200\,000$.

– **Subtask 4** (18 points) $N = 2$.

– **Subtask 5** (23 points) $N = 3$.

– **Subtask 6** (44 points) No additional limitations.

Examples

input	output
`2 42`	`3 41` `11 33`
`3 10`	`11 4 5` `1 13 6`

Explanation

In the **first sample case**, the sequences `3 - 41` (having sum 44 and XOR $K = 42$ as requested) and `11 - 33` (same sum, 44, and XOR $K = 42$ as requested) are a valid solution. The two sequences contain no duplicates and do not share numbers.

In the **second sample case**, a possible solution is constituted by the sequences `11 - 4 - 5` (with sum 20 and XOR $(11 \oplus 4) \oplus 5 = 15 \oplus 5 = 10$) and `1 - 13 - 6` (with sum 20 and XOR $(1 \oplus 13) \oplus 6 = 12 \oplus 6 = 10$).

Moving Heavy Loads (crane)

Everybody knows that Eng. Dog is the Messina's bridge manager: his duty is designing the new bridge over the Strait of Messina. One of the major challenges is to move a heavy load, the bridge's main span, from the base camp to the building site. Unfortunately, the payload is way too heavy to be moved by hand; therefore, he needs to operate a series of cranes.

The base camp is at considered to be at position 0, the destination is D meters away along a straight line and N cranes can be operated.

Figure 1: One of the N cranes of Eng. Dog.

The i-th crane is located at X_i meters from the base camp and has an arm R_i meters long. The crane can perform the following actions:

- move the head in $[X_i - R_i, X_i - 1]$ if it points to the left, or in $[X_i + 1, X_i + R_i]$ if it points to the right (extremes included);
- pick-up the load if it's directly under the head;
- put down the load directly under the head;
- rotate 180 degrees (since the area is densely populated, the load can be unloaded only along the straight line that connects the base to the destination).

Note that the i-th crane cannot pick-up or put down a load in position X_i, nor pass it through its base without rotating first! The crane can move and rotate even when it's carrying the load.

Operating the cranes can become pretty expensive. In particular, operating the i-th crane comes with these costs:

- moving the head (with or without a load) costs K_i euros for each meter of movement;
- rotating the arm (with or without a load) costs T_i euros for each rotation (which must be of 180 degrees).

Initially all heads are maximally extended and each crane may point to the left (towards the base camp) or to the right (towards the destination).

Our favourite engineer is pretty lazy: he really dislikes to walk from a crane to the next one. He only walks towards the destination, so the cranes can only be operated in increasing order of X_i.

Eng. Dog is interested on how much he has to pay to move the span from 0 to D: can you compute the cost for him?

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

Input

The first line contains 2 integers D and N, the distance from the base camp to the destination and the number of available cranes. The next N lines describe a crane each. Each line contains 5 integers:

- X_i, the position of the crane.
- R_i, the radius of the arm.
- T_i, the cost of turning 180 degrees the arm.
- K_i, the cost of moving the head of 1 meter.
- $\texttt{dir}_i$, the initial direction of the crane (0 means left, 1 means right).

Output

You need to write a single line with an integer: the minimum cost for moving the load from 0 to D.

Constraints

- $1 \leq D \leq 50\,000$.
- $1 \leq N \leq 500$.
- $0 \leq X_i \leq 100\,000$ for each $i = 0 \ldots N-1$.
- $1 \leq R_i \leq 500$ for each $i = 0 \ldots N-1$.
- $0 \leq T_i, K_i \leq 100\,000$ for each $i = 0 \ldots N-1$.
- $X_i < X_{i+1}$ for each $i = 0 \ldots N-2$.
- It is guaranteed that a solution always exists.

Scoring

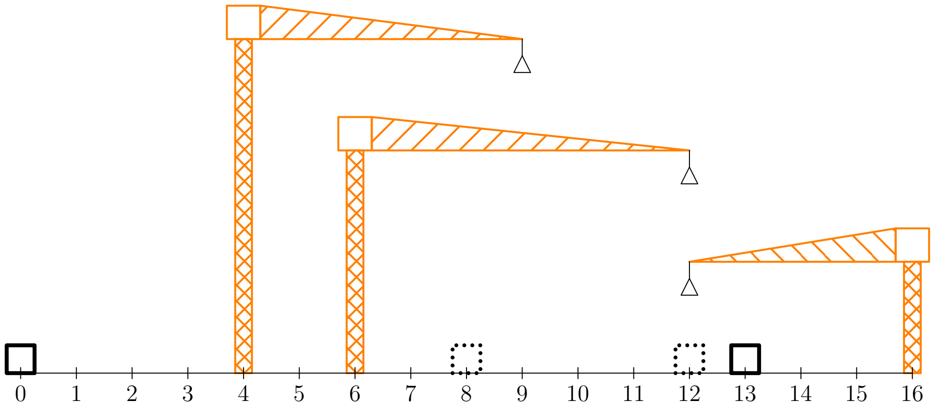
Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
- **Subtask 2** (8 points) $R_i = 1$ for each $i = 0 \ldots N-1$.
- **Subtask 3** (11 points) $N = 1$.
- **Subtask 4** (19 points) $D \leq 1000$, $T_i = 100\,000$, $K_i \leq 10$, $\texttt{dir}_i = 0$ for each $i = 0 \ldots N-1$.
- **Subtask 5** (12 points) $D \leq 10$, $N \leq 10$.
- **Subtask 6** (23 points) $D \leq 100$, $N \leq 100$.
- **Subtask 7** (18 points) $D \leq 1000$.
- **Subtask 8** (9 points) No additional limitations.

Examples

input	output
13 3 4 5 10 2 1 6 6 100 1 1 16 4 10 3 0	33
17 2 5 5 10 1 0 12 5 10 100 0	23

Explanation

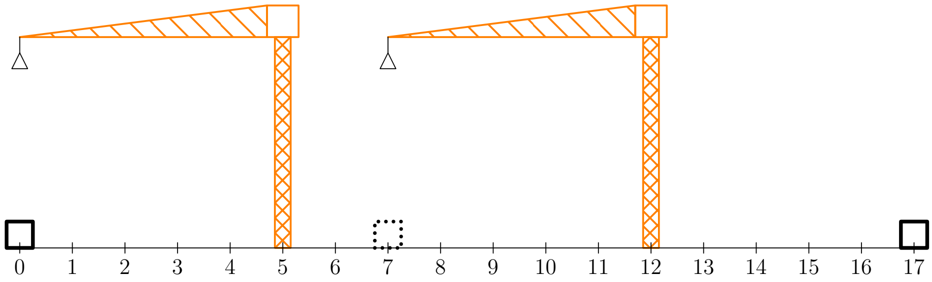


Representation of the initial state of the first sample case.

In the **first sample case** the optimal solution is:

- Rotate the first crane: pay 10 euro.
- Move the head from -1 to 0: pay 2 euro.
- Pick-up the package from 0.
- Rotate the crane: pay 10 euro.
- Unload the package at 8.
- Move the head of the second crane from 12 to 8: pay 4.
- Pick-up the package with the second crane.
- Move the head from 8 to 12: pay 4.
- Unload the package at 12.
- Load the package with the third crane.
- Move the head of the third crane from 12 to 13: pay 3.
- Unload the package at 13.

In total this solution costs $10 + 2 + 10 + 4 + 4 + 3 = 33$ euro. Another equivalent solution exists.



Representation of the initial state of the second sample case.

In the **second sample case** the solution is:

- Pick-up the package with the first crane from 0.
- Rotate the first crane: pay 10 euro.
- Move the head from 10 to 7: pay 3 euro.
- Unload the package at 7.
- Pick-up the page with the second crane.
- Rotate the crane: pay 10 euro.
- Unload the package at 17.

The overall cost is $10 + 3 + 10 = 23$ euro.

Virus Sequencing (dna)

Luca is working in an important analysis laboratory, and he is currently studying the various variants of the coronavirus. In particular, N different variants of the virus have been sequenced. That is, for each variant, its genome is described as an M characters long sequence S_i, composed only of zeros and ones.

Figure 1: The laboratory where Luca is working.

Luca needs to upload all these binary sequences to a central server as soon as possible, so that all the other laboratories around the world can access them.

For each sequence S_i, he can either upload the whole sequence or only the differences between S_i and another sequence that has already been uploaded. In the first case, he needs to upload M bits, in the second case he only needs to upload one bit for every position in which the two sequences differ.

What is the minimum number of bits he has to transfer in order to upload the genome of all the variants?

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

Input

The first line contains two integers N and M, respectively the number of variants and the length of the genome of each variant. Then N lines follow, the i-th of which containing one binary sequence S_i, that is the genome of the i-th variant.

Output

You need to write a single line with an integer: the minimum number of bits that Luca has to transfer.

Constraints

- $1 \leq N \leq 9\,000$.
- $1 \leq M \leq 100$
- S_i is a sequence composed of exactly M characters, each of them being either 0 and 1.
- $S_i \neq S_j$ for every $i \neq j$.

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** (14 points) $N \leq 9$, $M \leq 10$.
- **Subtask 3** (11 points) $N \leq 10$, $M \leq 25$.
- **Subtask 4** (15 points) $N \leq 20$, $M \leq 25$.
- **Subtask 5** (22 points) $N \leq 1000$, $M \leq 25$ and for each $i = 1 \dots N-1$ there exists $j = 0 \dots i-1$ such that S_i and S_j have exactly one different character.
- **Subtask 6** (31 points) $N \leq 1000$, $M \leq 25$.
- **Subtask 7** (7 points) No additional limitations.

Examples

input	output

`4 4`
`1111`
`0000`
`0011`
`1100`
	`10`

`2 4`
`1011`
`0100`
	`8`

Explanation

In the **first sample case**, Luca can do the following:

- Upload the first sequence, transferring $M = 4$ bits;
- Upload the differences between the third and the first sequence, transferring 2 bits;
- Upload the differences between the fourth and the first sequence, transferring 2 bits;
- Upload the differences between the second and the fourth sequence, transferring 2 bits;

He cannot do better than this and he has to transfer $4 + 2 + 2 + 2 = 10$ bits.

In the **second sample case** the two sequences never have the same character in the same position, so both of them must be fully uploaded, for a total of 8 bits.

Productivity Prize (productivity)

Marco has recently founded a startup company with N employees. When hiring, he promised to equally split a productivity prize of E euros across the best employees, if the company managed to get some conspicuous EU funds. Unexpectedly, the company was funded, and now he has to keep to his word!

Figure 1: Startup employees enjoying prizes.

However, Marco would very much prefer to award most of the E euros to himself. There is only one parameter he can leverage towards this end: the number of employees to award. He can choose any number K of employees, between a minimum of M (the employees directly involved in getting the funds) and a maximum of N (all employees). By choosing K wisely, some of the E euros may be left for him: since he promised to split the prize equally, he can always take the remainder of the division for himself!

For example, assume that $E = 100€$ and there are $N = 9$ employees, out of which $M = 7$ directly participated in the funding process. By choosing to award $K = 8$ employees, Marco will have to give 12€ to each of them, saving a whole 4€ for himself. Help Marco compute the greatest amount of remaining euros he can get by optimising K!

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

Input

The first and only line contains the three integers N, M, E.

Output

You need to write a single line with an integer: the maximum remainder of euros.

Constraints

- $1 \le M \le N \le 10^6$.
- $1 \le E \le 10^9$.

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) $N \le 1000$.

– **Subtask 3** (20 points) $N = M$.

– **Subtask 4** (50 points) No additional limitations.

Examples

input	output
9 7 100	4
4 2 42	2

Explanation

The **first sample case** is explained in the task statement.

In the **second sample case**, the best strategy is to choose $K = 4$.

Scheduling Services (scheduling)

Edoardo is preparing the infrastructure for the online contest of the current OIS edition's final round. As always, the contest will be managed using CMS, which is a platform composed of N services that all need to be executed in order for it to be able to work properly. To distribute the load, these services will be executed on *two* cloud servers.

Figure 1: The servers that will be used by Edoardo.

Edoardo needs to decide how to assign each service to one of the two servers. For the i-th service, it costs A_i euro to run it on the first server and B_i euro to execute it on the second one.

But there is more! Edoardo will also have to pay for the bandwidth used by services that need to communicate and that are assigned to different servers. This means that, if the i-th service is assigned to the first server and the j-th one to the second, then Edoardo will also have to pay C_{ij} euro.

What is the minimum amount of euro Edoardo will spend, if he assigns the services on the two servers optimally?

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

Input

The first line contains the only integer N, the number of services. The second line contains N integers A_i, the cost to run the i-th service on the first server. The third line contains N integers B_i, the cost to run the i-th service on the second server. Then N lines follow, the i-th of which contains N integers C_{ij}, the bandwidth cost Edoardo has to pay if the i-th and j-th services are assigned to different servers.

Output

You need to write a single line with an integer: the minimum amount of euro Edoardo will have to pay.

Constraints

- $1 \leq N \leq 200$.
- $0 \leq A_i, B_i, C_{ij} \leq 10^6$ for each $i = 0 \ldots N-1$, $j = 0 \ldots N-1$.
- $C_{ii} = 0$ for each $i = 0 \ldots N-1$.
- $C_{ij} = C_{ji}$ for each $i = 0 \ldots N-1$, $j = 0 \ldots N-1$.

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** (17 points) $N \leq 18$.

– **Subtask 3** (21 points) $C_{ij} = 0$ for each $i = 0 \ldots N-1$, $j = 0 \ldots N-1$.

– **Subtask 4** (28 points) $C_{ij} = 0$ if $|i-j| \neq 1$.

– **Subtask 5** (34 points) No additional limitations.

Examples

input	output
2 1 9 4 3 0 2 2 0	6
3 1 9 1 4 2 4 0 9 2 9 0 9 2 9 0	10

Explanation

In the **first sample case**, there are four ways to assign the services.

First Service	Second Service	Cost
First Server	First Server	$1+9=10$
Second Server	Second Server	$4+3=7$
First Server	Second Server	$1+3+2=6$
Second Server	First Server	$4+9+2=15$

In the **second sample case**, assigning all the services to the second server costs $4+2+4=10$ euro.

Una Lacrima Sul Visto (visa)

Giorgio has recently engaged with a foreign girl he matched online, and is about time to meet her in person! However, due to the pandemic, travelling around the globe is much more difficult than usual.

Figure 1: Giorgio gathering visas at the start of its trip.

The world is composed of N countries, numbered from 0 to $N-1$. In order to enter any country, even as a stop during a trip, it is mandatory to obtain a visa. In order to get the visa of country i, you need to pay V_i euros to convince them of your good intentions (and health). Every country i contains embassies of K_i other countries E_{ij} for $j = 0 \dots K_i - 1$, where you can buy the corresponding visas. Furthermore, every country i has direct connections with L_i other countries C_{ij} for $j = 0 \dots L_i - 1$, each with a travel cost of T_{ij} euros. The set of embassies and that of directly connected countries do **not** need to match in any particular way: there may be no embassy for a neighbouring country, as well as there may be embassies for countries that are not directly connected. Furthermore, direct connections do not have to be bidirectional (or have the same cost in both directions anyway).

Giorgio starts his journey from country 0 (European Union), with no visa for any country other than 0, and needs to reach country $N-1$ (Wakanda) where his new girlfriend lives, by taking direct connections and buying visas in embassies. Remember that he can never enter a country without having bought a visa in some other country he has previously visited! What is the best travel plan he can choose?

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

Input

The first line contains the only integer N, then N blocks of 4 lines each follow:

- The first line of block i contains integers V_i, K_i, L_i.
- The second line of block i contains K_i integers E_{ij}.
- The third line of block i contains L_i integers C_{ij}.
- The fourth line of block i contains L_i integers T_{ij}.

Output

You need to write two lines. The first line contains integer M, the length of the proposed travel plan. The second line contains M integers: the countries R_i that compose the travel plan.

Constraints

- $2 \leq N \leq 1000$.
- $1 \leq K_i, L_i \leq N - 1$ for each i.
- $0 \leq E_{ij}, C_{ij} \leq N - 1$ for each i and j.
- $0 \leq V_i, T_{ij} \leq 100\,000$ for each i and j.
- There are no duplicated embassies or direct connections from a same country.
- A country does not contain embassies of (or direct connections to) itself: $E_{ij} \neq i$, $C_{ij} \neq i$.
- A solution is guaranteed to exist.
- Test cases are random and not optimally difficult.

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) $L_i = 1$ for all i (countries are a path).

– **Subtask 3** (7 points) $K_0 = N - 1$ and $V_i = 0$ for all i (all visas are free in the EU).

– **Subtask 4** (8 points) $K_0 = N - 1$ (all visas are available in the EU).

– **Subtask 5** (9 points) $K_i = L_i$ and $E_{ij} = C_{ij}$ for all i and j (embassies and connections coincide).

– **Subtask 6** (19 points) $N \leq 5$.

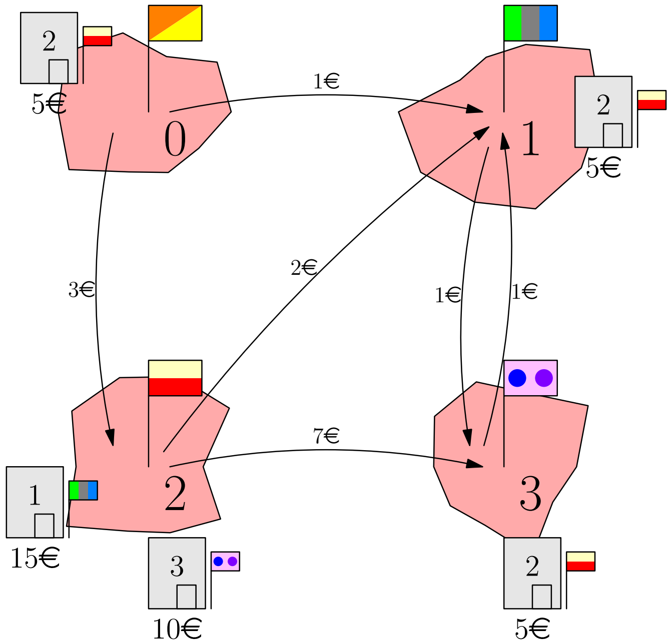
– **Subtask 7** (18 points) $N \leq 10$.

– **Subtask 8** (29 points) No additional limitations.

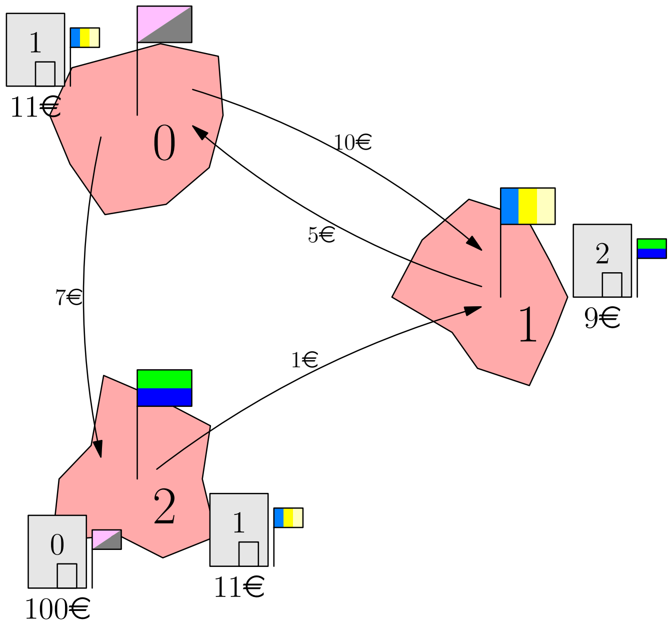
Examples

input	output
4 100 1 2 2 1 2 1 3 15 1 1 2 3 1 5 2 2 1 3 1 3 2 7 10 1 1 2 1 1	3 0 2 3
3 100 1 2 1 1 2 10 7 11 1 1 2 0 5 9 2 1 0 1 1 1	4 0 1 0 2

Explanation



In the **first sample case**, Giorgio must buy the visa for country 2 and travel there, paying $5 + 3 = 8$ euros. He has then two options: he can buy the visa for country 3 and travel directly there or he can buy the visas for both country 1 and 3, travel to country 1 and then go to the destination. In the first case he would pay $10 + 7 = 17$ euros, in the second case $15 + 10 + 2 + 1 = 28$ euros. So, the best path is $0 \to 2 \to 3$, for a total cost of $8 + 17 = 25$ euros.



In the **second sample case**, Giorgo can only do the following: buy the visa for the country 1 (11 euros), travel to country 1 (10 euros), buy the visa for country 2 (9 euros), travel to country 0 (5 euros) and then travel to country 2 (7 euros). The best path is then $0 \to 1 \to 0 \to 2$, with a total cost of $11 + 10 + 9 + 5 + 7 = 42$ euros.

Pronounceable Words (`words`)

William loves words. Today he is trying to come up with "pronounceable" words. What is a pronounceable word? Well, clearly "hello" is a pronounceable word, while "tqwukjt" is not. William defines a word as *pronounceable* if every one of its 3-character substrings is an *acceptable trigram*.

In the example above, the "wuk" substring could be reasonably pronounced so it's indeed an acceptable trigram, "qwu" is a bit ugly but it could still pass, but then what about "tqw" and "kjt"? They are just too weird to say!

To make things easy, William collected a list of all the K acceptable trigrams. With the hard part out of the way, now comes the easy stuff: generating the pronounceable words.

William is asking for your help. He left a note where he detailed how he wants you to write a program that reads in the *known trigrams* and two numbers N and L, and then performs the following operations:

1. Calculate a list of all the pronounceable words that are exactly L-characters long.
2. Sort the list alphabetically.
3. Finally, print the N-th word from the list.

Consider that there might be *a lot* of pronounceable words, and you're only required to print one word... It's OK if you don't actually compute all the words, no one needs to know.

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

Input

The first line contains three integers: N, L, and K. Each of the next K lines contains a 3-character string S_i.

Output

You need to write a single line with a L-letter long word: the N-th "pronounceable" word.

Constraints

- $1 \leq N \leq 10^{18}$.
- There are at least N pronounceable words.
- $3 \leq L \leq 10\,000$.
- $1 \leq K \leq 17\,576 = 26^3$.
- The length of S_i is exactly 3 characters, for each $i = 0 \ldots K-1$.
- S_i is always formed by lowercase alphabetic letters, and all trigrams are 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) $N = 1$, $L \leq 10$.

– **Subtask 3** (30 points) $N \leq 20$, $L \leq 10$.

– **Subtask 4** (30 points) $N \leq 1000$.

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

Examples

input	output

`3 4 4`
`aba`
`ada`
`dab`
`bad`
	`bada`

`52 10 5`
`aaa`
`baa`
`aba`
`abb`
`aab`
	`baaaaaaaab`

Explanation

In the **first sample case** there are exactly 4 pronounceable words that are 4 characters long: `abad`, `adab`, `bada`, `daba`.

In the **second sample case** there are exactly 73 pronounceable words that are 10 characters long.

Buy 3, Pay for 2 (3x2)

As his friends keep self inviting at his place, Luca is constantly looking for ways to save money when buying food.

Supermarkets are an excellent spot for this, with their never-ending promotional offers. In particular, “buy 3, pay for 2” is especially appealing when you need to buy a lot of things and you are certain that you will not waste a significant fraction of the goods.

Figure 1: "Buy 3, pay for 2" in an Italian fruit and vegetable shop.

The details of the promotions are rather simple: each person can use this offer only once, bringing three items to the cashier and skipping the payment of the less expensive one. If one buys less than three items, no particular discount applies.

Luca and his friends made a list of N goods they want to buy, along with their prices P_i. They are now heading towards the exit and they want to distribute the items among them so that the overall expense (that is, the sum of the amount each of them pays) is the least possible. How much will they pay?

> ☞ Among the attachments of this task you may find a template file `3x2.*` with a sample incomplete implementation.

Input

The first line contains the only integer N. The second line contains N integers P_i, the prices of the goods.

Output

You need to write a single line with an integer: the minimum total cost for buying all items, exploiting the promotion.

Constraints

- $1 \le N \le 300\,000$.
- $1 \le P_i \le 1000$ for each $i = 0 \ldots N-1$.
- You can assume there will always be enough friends to buy the goods in groups of (up to) three items.

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)	All prices P_i are equal.
– **Subtask 3** (30 points)	$N \le 10$.
– **Subtask 4** (25 points)	$N \le 10\,000$.
– **Subtask 5** (35 points)	No additional limitations.

Examples

input	output
`3` `5 34 8`	`42`
`4` `5 34 8 13`	`52`

Explanation

In the **first sample case** there are three items: only one person is needed to buy them. The "free" one will be the first (price 5), thus the overall expense is just $34 + 8 = 42$.

In the **second sample case** there are four items. With two friends, the best strategy is to assign to the first one the items that cost 34, 8 and 13, making her pay $34 + 13 = 47$. Then, the second friend has just to buy the remaining item at cost 5 (without any discount). The overall expense of the group to buy all the items is $47 + 5 = 52$.