

Circus Show (cannons)

The final of the Olympiads in Teams is finally here and William wants to celebrate this special occasion organizing a circus show! He does not have any experience in the sector though, thus he decided to contact experts from the *Organizing Improvised Shows* circus company. When Giorgio heard about the plan, he immediately started doubting the qualities of this group as they seem a bit... unorganized.

The main exhibition of the proposed show, which is also the most dangerous one, involves a *cannon man* who is repeatedly thrown by a series of cannons until he reaches the final destination. The group plans to have N cannons arranged in a line and start with the cannon man ready to be thrown by cannon 0; the final goal for him is to reach the last cannon $N - 1$.

Every cannon points to another cannon (potentially to itself) such that when the *cannon man* reaches the cannon i he is immediately thrown to the cannon T_i .



Figure 1: One of the cannons used for the show.

While this sounds like a lot of fun, Giorgio was right: folks at *Organizing Improvised Shows* are very unorganized. After placing the cannons, they lost part of the original plan for setting the directions and now is unclear the trajectory the *cannon man* will take.

Giorgio wants to take charge of the situation to do a last-minute fix. After studying the current setting for every cannon, he is devising a plan to change some of them in order to make sure the cannon man is able to reach the final destination. Changing the setting for a cannon consists in modifying the value of T_i for that cannon and is known that modifying it to another value T_i' will cost $|T_i' - i|$ units of time.

The show is about to start and Giorgio is running out of time: help him find the *minimum* total time required to change the setting of some of the cannons to let the cannon man reach cannon $N - 1$ from cannon 0.

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

Input

The first line contains the only integer N , the number of cannons. The second line contains N integers T_i : the i -th one contains the index of the cannon to which the cannon man will be thrown if he lands on cannon i .

Output







You need to write a single line with an integer: the minimum total time required for making the changes so that the cannon man is able to reach cannon $N - 1$.

Constraints

- $1 \leq N \leq 1\,000\,000$.
- $0 \leq T_i < N$ for each $i = 0 \dots N - 1$.

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.

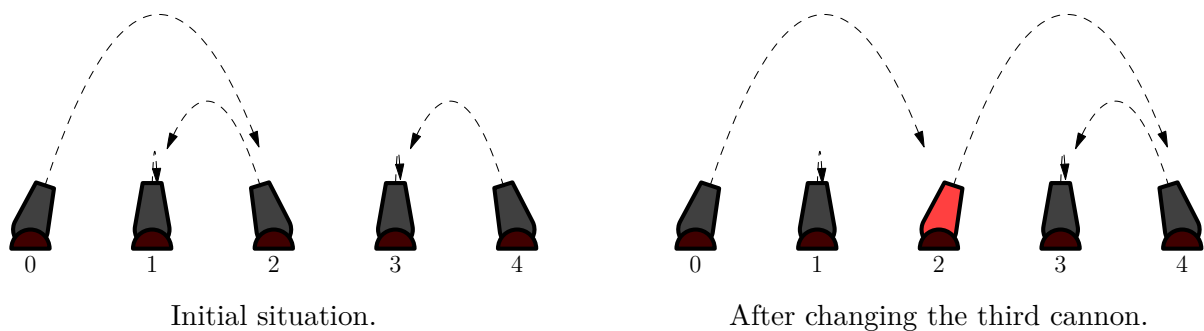
- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $N \leq 10$.

- **Subtask 3** (10 points) It is guaranteed that exists a solution the cost of which is *at most* one unit of time.

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

- **Subtask 5** (20 points) $N \leq 7000$.

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


Examples

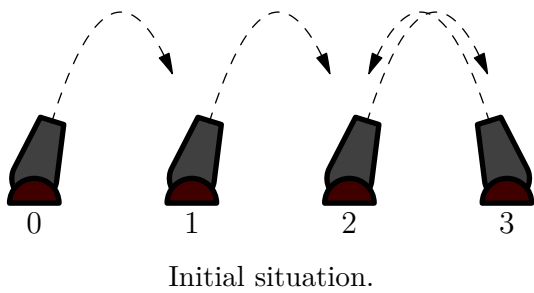
input	output
5 2 1 1 3 3	2
4 1 2 3 2	0

Explanation

In the **first sample case** a possible solution is to modify the setting of the third cannon, making it pointing directly towards the final destination. This only change has a cost of $4 - 2 = 2$ units of time.



In the **second sample case** the cannon man is already able to reach the final cannon without any modification to the current setting.



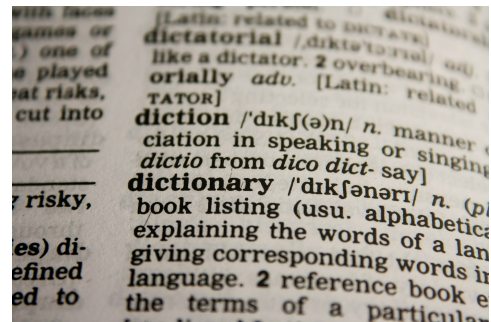
Multi-Layer Dictionary (dictionary)

When William encounters a new English word, he usually looks for its definition in the dictionary. For example, a definition for the word “unencumbered” is as follows:

unencumbered = not having any burden

Sometimes this process creates a recursion, because William might then need to look up the definition for “burden”, which again might be using new words that William doesn’t know yet.

To avoid encountering this issue, you should know that every dictionary has a *defining vocabulary*: a fixed list of words that was used to write the dictionary’s definitions. For example, the *Longman Dictionary of Contemporary English* uses a defining vocabulary of approximately 2000 words, while the *Oxford Advanced Learner’s Dictionary* uses one of approximately 3000 words. If you learn all of those beforehand then you can safely read the dictionary without encountering unexpected words.



Learning a couple thousand words just to be able to use a dictionary is too much to ask of William: being lazy, he always looks for an easier way. In fact, he found a (freely available online) dictionary called **Learn These Words First**¹. This dictionary is special because it’s *layered*:

- There is a first layer that contains just 61 words (called *primitive words*) which are mostly universal concepts, explained through pictures.
- Each successive layer builds upon the previous layers by defining some new words using *only the words that are already defined in the previous layers*.

This means that by reading the dictionary in the correct order (level by level) you will never encounter unexpected words, and instead of thousands of primitive words you only need to learn 61 of them!

The discovery of this dictionary led William to wonder about a challenging new problem: given a dictionary, find the smallest possible set of primitive words that you need to know in order to be able to recursively define every other word of the dictionary (either by using the initial words or by using words derived from other definitions).

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

Input

The first line contains the integer N , the *total number of distinct words* used in the dictionary (including the words being defined). The second line contains the integer D , the number of definitions in the dictionary. Each of the next D lines contains one definition of a word: each definition starts with the word being defined, then an integer K , and then K words that form its definition.

¹<http://learnthesekeywordsfirst.com/>

Output

You need to write a line with an integer W , as small as you can, followed by W lines with words able to define all other words (in any order).

Constraints

- $1 \leq D \leq N \leq 1000$.
- Each word uses between 1 and 20 lowercase alphabetical letters.
- Some words in definitions may not have a definition themselves: you need to learn them first!

Scoring

This is a **partial score** task. Your program will be tested against several testcases and it will receive a score based on whether your proposed solution is valid and minimal. There will be 10 testcases (divided in 10 different subtasks, so the score of each testcase is independent from the others, and the max score for each testcase will be taken over all submissions). The first subtask contains the sample testcases, so in total there are 11 subtasks. For each testcase, the score will be:

- 0 if the proposed solution is not valid (i.e. if the output is malformed or if the selected words are not enough to recursively define all of the dictionary's words).
- $\min \left\{ 1, 0.2 + 0.8 \left(\frac{N-W}{N-W_{ref}} \right)^3 \right\}$ where W_{ref} is a reference result for that input.

Examples

input	output
7 3 cat 3 not a dog dog 3 a domesticated mammal human 2 a mammal	4 not a domesticated mammal
3 2 easy 2 not difficult difficult 2 not easy	2 not easy
4 4 a 1 b b 1 c c 1 d d 1 a	1 c

Explanation

In the **first sample case** by learning those 4 words we can define “human” and “dog”. After that, since we know the word “dog”, we can define “cat” as well (without increasing the set of initial words).

In the **second sample case** there are two possible solutions: “not” + “easy”, or “not” + “difficult”.

In the **third sample case** knowing *any* word is enough.

Pay That Box Again! (gameshow2)


After William's modest success at the *Pay That Box!* game show, everybody in the OIS staff tried to get selected for it as well... and Edoardo made it! Since he doesn't like to take blind decisions, he also managed to hack into the show servers and downloaded the list of boxes that he will encounter.



Figure 1: Boxes, more boxes, and boxes again.

As you already know, Edoardo will choose a starting budget of M euros, then open a sequence of N boxes one at a time, from $i = 0$ to $N - 1$. Each box contains a prize, which is worth V_i euros for Edoardo, and he can either *get* it using P_i euros from his budget, or he can *pass* it gaining a compensation of C_i euros.

At the end of the show, Edoardo will evaluate his *gain* G as the sum of values V_i of the prizes bought plus the (possibly negative) money overall gained in the process. Of course, he wants to have the highest gain G possible, using the smallest budget M allowing him to get it. Help him get the best deal!

 Among the attachments of this task you may find a template file `gameshow2.*` with a sample incomplete implementation.

Input

The first line contains the only integer N . The second line contains N integers V_i . The third line contains N integers P_i . The fourth line contains N integers C_i .

Output






You need to write a single line with two integers G and M : the highest gain and the smallest budget allowing it.

Constraints

- $1 \leq N \leq 100\,000$.
- $0 \leq V_i, P_i, C_i \leq 10\,000$ for each $i = 0 \dots N - 1$.
- The starting budget M chosen by Edoardo must be at least zero.

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) $N \leq 100$.

- **Subtask 3** (25 points) $P_i = 0$ for each i .

- **Subtask 4** (30 points) $C_i = 0$ for each i .

- **Subtask 5** (25 points) No additional limitations.


Examples

input	output
4 42 30 71 69 23 15 30 62 12 20 37 15	95 33
8 30 38 99 88 75 32 10 76 60 36 32 31 32 18 45 16 21 4 35 34 21 23 6 19	281 70

Explanation

In the **first sample case**, starting with a budget of 33 euros, Edoardo can get the first prize (remaining with 10 euros), pass the second (recovering to 30 euros), get the third consuming all of his budget, and finally pass on the last (ending with 15 euros). Overall, Edoardo gained prizes for a total of $42 + 71 = 113$ euros, consuming $33 - 15 = 18$ euros, so his total gain is $113 - 18 = 95$ euros. Edoardo cannot get a larger gain with a higher budget, and also cannot get the same gain with a lower budget: if starting with 32 euros or less, his best option would be to get the third prize only for a total gain of 83 euros.

In the **second sample case** the best strategy is getting for the third, fourth, fifth and last prizes and pass the rest.

Splitting the Bill (money)

Giorgio and his friends like going out to have dinner a lot. Being a big group of N friends, including Giorgio, they always call in advance to book a table. However, every time they have to split the bill they don't know how to do it, since all of them only have credit cards! This means that there is always someone that pays the dinner for everybody, lending money to the others.



Figure 1: Giorgio and his friends deciding who has to pay.

Giorgio and his friends would like to make things even. They recorded M lendings of money, the i -th of which being A_i lending W_i euros to B_i . They need to decide which money transfers each of them needs to do, without making too many of them. In particular they want to make at most $N - 1$ money transfer in total. Can you help Giorgio and his friends find which money transfers need to be done?

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

Input

The first line contains two integers N and M , the number of friends and the number of money lendings. The following M lines contain three integers each, A_i , B_i and W_i , meaning that A_i lent W_i euros to B_i .

Output

In the first line you need to write K , the number of money transfers to be made in order to make things even; K must be less or equal to $N - 1$. In the following K lines you need to describe the j -th money transfer writing three integers: the person making the money transfer, the person receiving it and the amount of money transferred. The amount of money in a single money transfer must be at least one and at most 10^9 .







If there are multiple solutions, print any of them.

Constraints

- $1 \leq N \leq 200\,000$.
- $1 \leq M \leq 200\,000$.
- $0 \leq A_i, B_i < N$ for each $i = 0 \dots M - 1$.
- $1 \leq W_i < 5000$.
- $A_i \neq B_i$, for each $i = 0 \dots M - 1$.
- 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** (10 points) $M < N$.

- **Subtask 3** (15 points) $A_i = A_j$ for each $i = 0 \dots M - 1, j = 0 \dots M - 1$.

- **Subtask 4** (15 points) $B_i = B_j$ for each $i = 0 \dots M - 1, j = 0 \dots M - 1$.

- **Subtask 5** (25 points) $N \leq 100, M \leq 10\,000$.

- **Subtask 6** (35 points) No additional limitations.


Examples

input	output
4 3 0 2 3 0 3 4 3 1 2	3 1 3 2 2 0 3 3 0 4
4 5 3 0 1 0 1 2 1 2 1 2 3 1 3 0 1	2 1 0 1 0 3 1

Explanation

In the **first sample case**, at most $N - 1 = 3$ transfers can be made. All the friends are happy because they gave as much as they received.

In the **second sample case**, we can see that the first four lendings can be fixed by having friend 1 send to 0 a total of one euro. Then we revert the last one. Note that this is not the only possible solution.

Flood Forecasting (rainstorm)

Marco was recently hired by the Italian road-safety authority to oversee the rain storm forecasting operations. He was provided with a map of M bidirectional roads that connect N cities.

We know that, if there is no rain, we can traverse every road and it is always possible to reach any city starting from any other city. However, as the expected rain flooding (in millimeters) becomes higher, some of the roads become impossible to traverse and have to be closed to traffic. For each road, Marco knows exactly how many millimeters of rain flooding it can withstand at most.



As long as there is at least one way to reach every city starting from any other city (even if it means taking a longer-than-usual route), life can go on pretty much as usual. However, if the flooding causes any two cities to be disconnected, a state of emergency must be declared!

Help Marco calculate what is the maximum amount of rain flooding (in millimeters) that the whole network can withstand: that is, the highest amount of rain flooding that does not force the road-safety authority to declare a state of emergency and stop the traffic between any two cities.

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

Input

The first line contains two integers N and M . Each of next M lines contains three integers: A_i, B_i, R_i . These integers indicate that there is a road connecting city A_i to city B_i which can be traversed as long as the rain flooding level does not exceed R_i millimeters.

Output






You need to write a single line with an integer: the maximum amount of rain flooding that the network can withstand.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq M \leq 100\,000$.
- $0 \leq A[i], B[i] \leq N - 1$, $A[i] \neq B[i]$ for each $i = 0, \dots, M - 1$.
- $1 \leq R[i] \leq 100\,000$ for each $i = 0, \dots, M - 1$.
- If there is no rain, it's always possible to reach any city from any other city.

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.

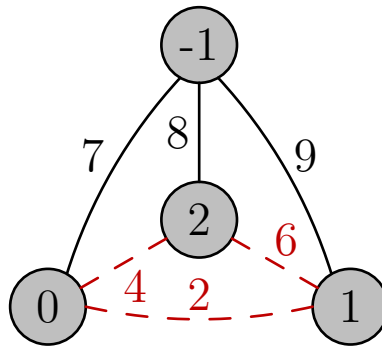
- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $M = N - 1$.

- **Subtask 3** (20 points) $N \leq 10$, $M \leq 15$.

- **Subtask 4** (50 points) $N \leq 1000$, $M \leq 2000$.

- **Subtask 5** (20 points) No additional limitations.


Examples

input	output
4 6 2 0 9 0 1 7 2 1 2 0 3 8 2 3 6 1 3 4	7
7 6 4 3 9 2 3 7 1 5 6 5 0 3 6 0 5 0 2 10	3

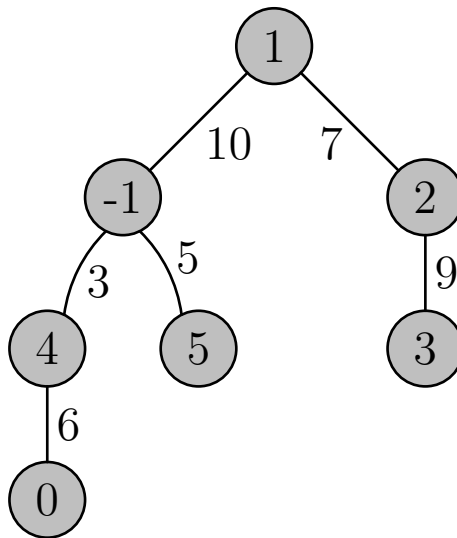
Explanation

In the **first sample case**, when the rain flood level reaches 7 there are only three city connections that are still usable (with $R_i = 7, 8, 9$), and these three connections are enough to reach any city starting from any other city.



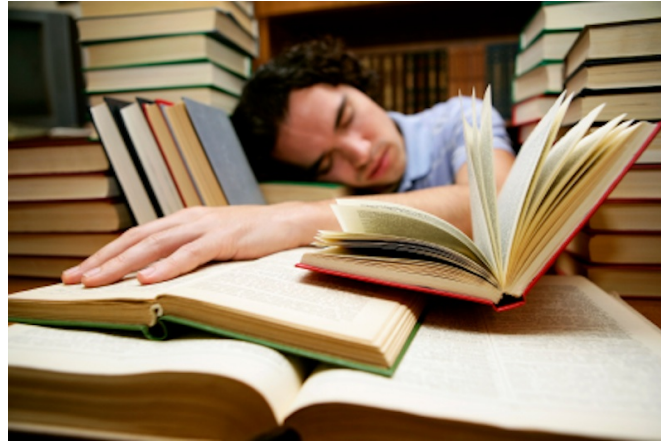
If the flood level reaches 8, city 0 gets disconnected from the rest of the network, thus 7 is the maximum flood level that the network can withstand.

In the **second sample case**, losing any connection makes some city unreachable from some other city. Thus 3 is the maximum flood level sustained.



Exam Session (studyplan)

Marco has been studying at *Politecnico di Milano* for many years, he already tackled many exam sessions, but this year he has lots of hard exams so he decided to organize his study plan a bit better.



He wants to prepare all the N exams he has, but noticed that not all of them are equally hard, so he estimated the number of hours H_i needed for each exam. Having followed all the lessons he also noticed that it's possible that, knowing the topics of an exam, some other exams become much easier; because of that he wants to study the subjects according to some partial ordering². For example studying *Calculus I* should be studied before *Calculus II* and *Geometry*, but those two can be studied in any order.

Marco is very multitask, so much that if he completed all the prerequisites of a subject he *can* start right away studying that subject, even if he's already studying something else! Furthermore, studying more than a subject in parallel doesn't increase the time required to complete them: if the i -th subject starts at time t , it will end at time $t + H_i$ regardless of how many other subjects Marco is studying in parallel.

After a quick calculation, Marco already knows the minimum time required to complete all the subjects... but he also wants to know when he can take a break! He allows himself to delay the start of only one subject, but he doesn't want to delay the total time required to study all the subjects.

Help Marco finding, for each subject, which is the latest time he can start studying that subject without extending the total required time.

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

Input

The first line contains the only integer N . The next line contains N integers separated by space: the values of H_i .

The following N lines describe a subject each, they are in the following format:

- First an integer k_i : the number of prerequisites of the i -th subject.
- Then k_i integers: the index of the prerequisites of the i -th subject.

²A partial order defines the ordering between *some* pair of items, while a *total* order is defined between all pairs.

Output

You need to write a single line with N integers: the latest time the i -th subject can start without delaying the completion of the study plan.


Constraints

- $1 \leq N \leq 100\,000$.
- $\sum_{i=0}^{N-1} k_i \leq 500\,000$.
- $1 \leq H_i \leq 10\,000$ for each $i = 0 \dots N - 1$.
- It's always possible to devise a study plan.


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)




$\sum_{i=0}^{N-1} k_i = N - 1$, $k_i \leq 1$ for each $i = 0 \dots N - 1$ and every subject is a direct prerequisite of at most one other subject.
- Subtask 3 (10 points)




There are no prerequisites between the subjects.
- Subtask 4 (20 points)



$N \leq 10$ and $H_i \leq 10$.
- Subtask 5 (35 points)



$N \leq 1000$ and $\sum_{i=0}^{N-1} k_i \leq 5000$.
- Subtask 6 (25 points)



No additional limitations.

Examples

input	output
5 10 5 30 15 10 0 1 0 1 0 1 1 2 2 3	0 20 10 25 40
4 15 7 3 10 0 1 2 1 0 0	0 18 15 15

Explanation

In the **first sample case** there are 5 subjects, the subject 0 has no prerequisites, so it can start at time $t = 0$. When it ends at time $t = 10$ subjects 1 and 2 can start, the first ends at $t = 15$ so subject 3 can start. Subject 4 has to wait for subjects 2 and 3: it can start at $t = 40$, and after it ends all the subjects have been completed at $t = 50$.

Marco can delay some subjects without exceeding the total time of 50, for example if subject 1 starts at $t = 20$ instead of $t = 10$, subject 3 starts at $t = 25$ but subject 4 is not delayed, so the total time is still 50. On the other hand it's not possible to delay subjects 0, 2 and 4 without exceeding the minimum required time of 50.

In the **second sample case** subjects 0 and 4 can start right away, after $t = 15$ subject 2 can start and at $t = 18$ subject 1 starts. When subjects 1 and 4 end the plan is completed, at $t = 25$.

The only subject that can be delayed is 4, and it can start at $t = 15$ without delaying the study plan.

Mysterious Sum (sum)

Luca is going to Bologna by train to attend the grand final of OIS's current edition, but a recent accident blocked the high-speed railway and made the journey way longer. To kill time, Luca loves to solve some puzzles and he is now working on an interesting one.

In this kind of puzzle, you are given a simple sum of two integers numbers and the result of this addition, but there is one caveat. Operands' digits have been substituted with letters, like in the next example, following a rule commonly known as “same digit, same letter”.

$$\begin{array}{r} A B + \\ C A = \\ 58 \end{array}$$

More precisely, this means that every digit from 0 to 9 has been linked to an uppercase letter from A to J and every occurrence has been substituted with that letter.

Moreover, the mapping always links different digits to different letters and vice versa, so that no two digits have the same associated letter and that no two letters are associated with the same digit.

Luca is still a bit puzzled and his train ride is almost over: can you recover the two addends for him?

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

Input

The first two lines contain respectively the two addends with digits substituted with letters. Both the addends have the same length N . The third line contains the result given by the sum of the two operands.

Output

You need to write two lines which contain respectively the digits of the first and the second addend so that their sum equals the result given in input.

Constraints






- $1 \leq N \leq 10\,000$.
- Addends are written as valid numbers: they *never* have leading zeros (unless that the whole number is just a 0). To exemplify, 000 and 012 are not valid numbers, while 0 and 123 are.
- It is guaranteed that a solution exists.
- If there are multiple solutions, you can output any of them.



Figure 1: “La settimana enigmistica” is a very popular Italian weekly publication. This is the first issue, published on 23rd January 1932!

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** (30 points) Summing the two operands *never* involves a carry.
 
- **Subtask 4** (40 points) $N \leq 50$.
 
- **Subtask 5** (20 points) No additional limitations.
 

Examples

input	output
AB CA 58	35 23
AABCDEF DBGJCIH 12345678	7752410 4593268

Explanation

In the **first sample case** a possible solution is to map 3 to A, 5 to B and 2 to C (we do not care about the remaining digits). This leads to 35 as the first addend and 23 as the second one, their sum is correctly 58. Other valid solutions may exist (but note that assigning 0 to A or C would produce invalid numbers).

In the **second sample case** a mapping that produces the correct results is the following.

A	B	C	D	E	F	G	H	I	J
7	5	2	4	1	0	9	8	6	3

Graduation Card (text)


Luca has been invited to Edoardo's graduation party, where he knows there will be a lot of confetti! He bought a graduation card and now he needs to write a little congratulatory message on it. The message he wants to write is N words long, where the i -th word is W_i .



Figure 1: The graduation card that Luca bought.

Luca knows that he always makes mistakes when writing on little cards. In particular he always overestimates the space he can use to write the message, so he ends up writing the last words much smaller than the first ones. To prevent this, he has decided that each line must contain at most K letters or spaces.

He wants to lay out the message so that the first word is at the beginning of the first line, followed by all the other words. When a word doesn't fit in a line, that word is put on a new one, followed by the other words, and so on. Can you help Luca lay out the message?

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

Input

The first line contains the integer N , the number of words in the message and K , the maximum length of the line, including the spaces. The following N lines contain one string W_i , the i -th word of the message.

Output

You need to write the message in such a way that each line has at most K characters or spaces, as described above. Between each consecutive word in the same line there must be one space character. Spaces at the end of a line will be ignored.


Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq K \leq 10^9$.
- $1 \leq |W_i| \leq \min(20, K)$, for each $i = 0 \dots N - 1$, where $|W_i|$ is the length of the i -th word.
- W_i is composed only of lowercase letters of the English alphabet, for each $i = 0 \dots N - 1$.


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)




$N \leq 3$.
- Subtask 3 (20 points)



$|W_i| = |W_j|$ for each $i = 0 \dots N - 1, j = 0 \dots N - 1$.
- Subtask 4 (30 points)



$N \leq 1000$.
- Subtask 5 (40 points)



No additional limitations.

Examples

input	output
5 8 this is a sample message	this is a sample message
4 10 congrats for your graduation	congrats for your graduation

Explanation

In the **first sample case**, note that the first three words have a cumulative length of $4 + 2 + 1 = 7$ characters, which is less than $K = 8$. However also the spaces between consecutive words matter, so it is not possible to fit the word “a” in the first line (as the line would have $4 + 1 + 2 + 1 + 1 = 9$ characters and spaces).