

Bitcoin trading (bitcoin)

Near the end of 2017, the cryptocurrency known as “Bitcoin” (BTC) reached a record value of more than 16 000€. William noticed this increase and is now wondering: how much money could someone have made in the *best possible scenario* starting with 1 BTC?

For this reason, William collected every price increase (and decrease) of the last N days, and now he wants to know: starting with 1 BTC, how much € could he get by repeatedly selling and buying?



Figure 1: The graph of Bitcoin price changes.

For example, let’s assume the value of 1 BTC is initially 1000€ and that there will be 5 price changes:

$$-20, \quad +85, \quad +10, \quad -100, \quad +1$$

In this case, the best strategy is to:

- Sell 1 BTC immediately, earning 1000€.
- Wait for the price to go down by 20€, and then sell the EUR for BTC.
- Wait for the price to go up by 85€, and hold the BTC.
- Wait for the price to go up by 10€, and sell the BTC for EUR.
- Wait for the price to go down by 100€, and then sell the EUR for BTC.
- Wait for the price to go up by 1€, and then sell the BTC for EUR.

In this way one would end up with a little more than 1098€ in their pockets!

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



Input

The first line contains two integers N and E , respectively the number of days for which William took note of the price change and the initial price (in euros) of 1 BTC. The second line contains N signed integers C_i , the price change for each day.

Output

You need to write a single line with a real value: the maximum possible value in euros achievable at the end of the N days. The answer will be considered correct if the absolute or relative error is lower than 10^{-6} .

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq E \leq 10\,000$.
- $-100 \leq C_i \leq 100$ for each $i = 0 \dots N - 1$.
- We can assume that the price changes **never** cause the price of BTC to go lower than 1€, that is: $E + C_0 + C_1 + \dots + C_i \geq 1$ for any i .

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]: The price always increases.
- **Subtask 3** [30 points]: $N \leq 10$.
- **Subtask 4** [30 points]: $N \leq 100$.
- **Subtask 5** [20 points]: No additional limitations.

Examples

| input.txt | output.txt |
|-------------------------------|---------------|
| 5 1000 -20 +85 +10 -100 +1 | 1098.06384092 |
| 5 1000 +10 +11 +12 +13 +14 | 1060.00000000 |

Explanation

The **first sample case** is the example discussed before.

In the **second sample case** the price is only going up: the best strategy is to buy early and sell late.

Metro platforms (metro)

Since he began his university studies in Milan, Edoardo started commuting by metro everyday. He usually goes to class by metro, but today he is so sleepy that he is not sure from which metro stop he started, nor which direction he took!

It's worth to notice that trains in metro stations don't always open their doors from the same side: sometimes you need to get off from the left side, sometimes from the right side. There is a small map inside the train, which shows the position of the platform (left or right exit). With this information, Edoardo believes he can easily understand from which metro stop he started his commute.

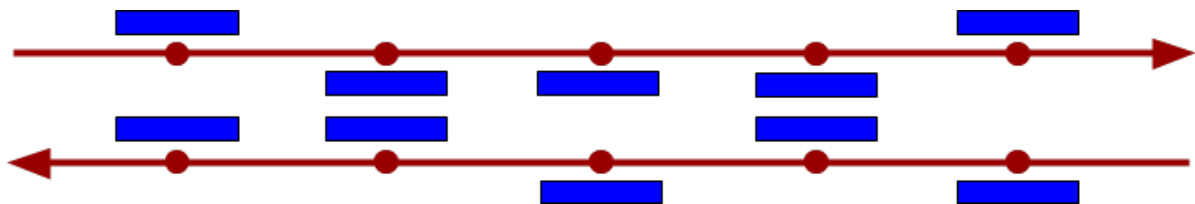
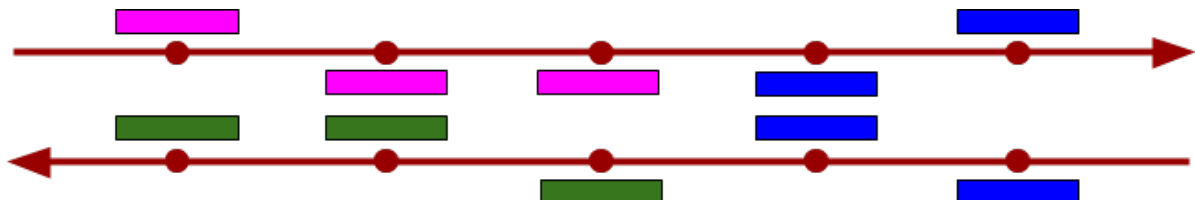


Figure 1: The small map inside the train, showing the location of the platform.

Edoardo paid attention during the last M stops, and for each stop he noted on which side the train opened its doors. Help Edoardo count how many possible starting metro stops are compatible with his observations.

For example let's say that Edoardo remembers that, during the last 3 stops, the train opened its doors on the left side, then right side, then right side again. Using the map above, we can see that this observations are compatible with just 2 stops:



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

Input

The first line contains a single integer N , the number of stops in the whole metro line. The second and third lines contain N characters each, respectively the description (seen from above, east to west) of where the doors will open for the “forward” and the “backward” metro lines. The next line contains a single integer M , the number of observations made by Edoardo. The last line contains M characters, the description of each observation.

Output

You need to write a single line with an integer: the number of possible starting metro stops in one way or the other.

Constraints

- $1 \leq M \leq N \leq 10\,000\,000$.
- The characters in the descriptions are just \wedge or v , respectively: the door will open on the left / right (when going east), and right / left (when going west).
- The characters in the observations are just $<$ or $>$, respectively: the door opened on the left / right.

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]: The doors always open on the left.
- **Subtask 3** [30 points]: $N \leq 100$.
- **Subtask 4** [40 points]: $N \leq 10\,000$.
- **Subtask 5** [10 points]: No additional limitations.

Examples

| input.txt | output.txt |
|---|--------------|
| <pre>5 ^vvv^ ^^v^v 3 <>></pre> | <pre>2</pre> |
| <pre>5 ^^^^^ vvvvv 3 <<<</pre> | <pre>6</pre> |

Explanation

The **first sample case** is the example discussed before.

In the **second sample case** the doors always open on the left, so every contiguous sequence of 3 stops identifies a valid starting metro stop.

Circular monuments (monuments)

William is a famous urban planner and he has been commissioned the design of a new high road to connect Pordenone to Campobasso. There is only a small problem: he has to build the road across a park where there are N very important statues. These statues are placed along a circumference at distinct integer angles A_i (from 0 to $2K - 1$).



The road *must* cross the center of the circle (like a diameter) and it is wide such that, if it starts at angle p , it covers all the statues between p and $p + L - 1$ and between $p + K$ and $p + K + L - 1$ (extremes included).

William can design the road at any angle, but he wants to find the minimum number of ancient monuments he is forced to demolish in order to build it.

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

Input

The first line contains integers N, K, L . The second line contains N integers A_i .

Output

You need to write a single line with an integer: the minimum number of statues to demolish.

Constraints

- $1 \leq K \leq 10^9$.
- $0 \leq N \leq 1\,000\,000$.
- $1 \leq L \leq K$.
- $0 \leq A_i \leq 2K - 1$ for each $i = 0 \dots N - 1$.
- All the angles A_i are unique, thus $N \leq 2K$.

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]: $L = 1$.
- **Subtask 3** [10 points]: $K \leq 100$.
- **Subtask 4** [40 points]: $A_i < K$ for all $i = 0 \dots N - 1$ and $K \leq 1\,000$.
- **Subtask 5** [20 points]: $K \leq 1\,000$.
- **Subtask 6** [20 points]: No additional limitations.

Examples

| input.txt | output.txt |
|-----------------------|------------|
| 6 6 2 0 1 6 9 3 4 | 1 |
| 6 8 2 8 0 1 6 15 4 | 0 |

Explanation

In the **first sample case** the road can be built at $p = 1$ (and cover angles 1, 2, 7 and 8) or at $p = 4$ (and cover angles 4, 5, 10 and 11); in both cases taking down only one statue.

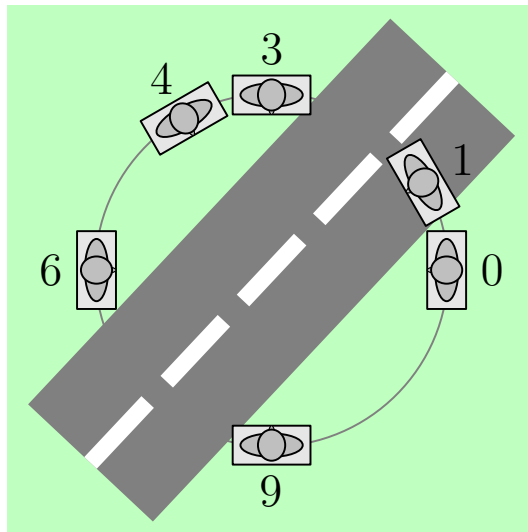


Figure 1: The situation in the first example.

In the **second sample case** the road can be build without touching any monument at $p = 2$ (covering angles 2, 3, 10 and 11).

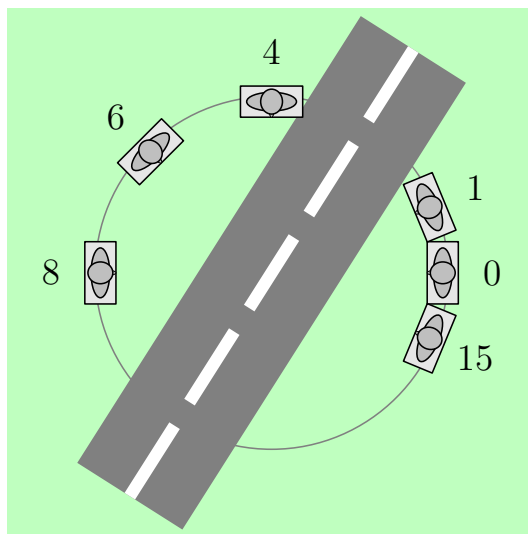


Figure 2: The situation in the second example.

Password generator (passwords)

Edoardo has decided to improve the current password generator for the OIS, in order to obtain passwords that are even more secure and easy to remember at the same time!



Figure 1: Failed attempts to find passwords that are both secure and easy to remember.

Of course, this is not an easy task. He decided that the best way to generate such a password (with respect to a given language consisting of N words W_i for $i = 0 \dots N - 1$) is to find a string P of length L such that:

- **consistency:** every substring of P of length K appears in some word of the given language;
- **novelty:** every substring of P of length $K + 1$ does not appear in any word of the given language.

Help Edoardo find such a word!

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

Input

The first line contains integers N, K, L . The following N lines contain the strings W_i .

Output

You need to write a single line with a string: a password that is novel and consistent.

Constraints

- $1 \leq K < 10$.
- $K < L \leq N \leq 10\,000$.
- Each word W_i consists only in lowercase alphabetic letters ‘a’ — ‘z’ and has length at most 10.
- It is guaranteed that at least one novel and consistent password exists. When there are many of them, you can print any.

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** [40 points]: $K = 1, N \leq 100$.
- **Subtask 3** [30 points]: $K = 1$.
- **Subtask 4** [20 points]: $N \leq 100$.
- **Subtask 5** [10 points]: No additional limitations.

Examples

| input.txt | output.txt |
|--|------------|
| 5 1 4 babbo bocca boo baobab bacco | bcbbc |
| 7 2 6 mamma amare reame mamma mamma reame amare | mmmear |

Explanation

In the **first sample case**, the novel and consistent passwords are `aaaa`, `bcbbc`, `cbcb` and `oaaa` since they have existing single letters, but non-existing pair of letters.

In the **second sample case**, possible passwords are `mmnear`, `mmmea`, `mmmmme` and `mmmmmm`.

Keylogger (pin)

Luca loves IT security, but he is a bit worried about the increasing number of attacks targeting smartphones.

Recently, a number of smartphone companies have been discovered putting a sort of “keylogger” which secretly writes everything that is typed to a hidden file. Without much surprise, Luca found that even his smartphone does this thing and wants to understand the security implication of this absurd behaviour.

He has gathered the log file from the phone in order to see whether he is able to recover the PIN (this would be a *huge* security issue!). Luca has stripped away all non-digit characters from the file, leaving a sequence of N digits.

Under the assumption that the PIN consists of K digits and that the PIN is frequently typed, what is the sequence of four digits that is more likely to be Luca’s PIN?

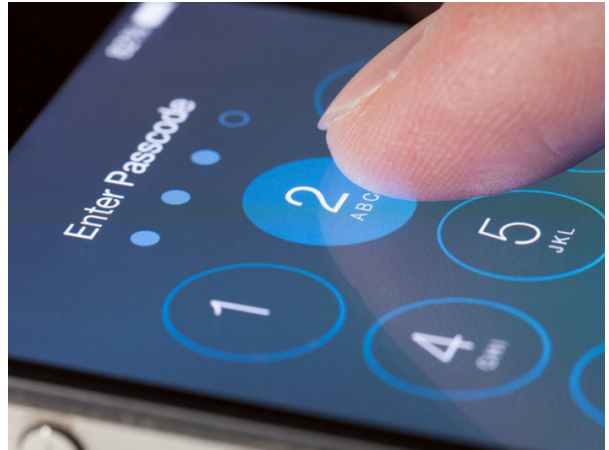


Figure 1: A user entering their 4-digit PIN.

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

Input

The first line contains integers N and K . The second line contains N integers D_i , the digits that appear in the log file.

Output

You need to write a single line with K digits, separated by a space, which constitute Luca’s PIN.

Constraints

- $4 \leq N \leq 1\,000\,000$.
- $4 \leq K \leq 9$.
- $0 \leq D_i \leq 9$ for each $i = 0 \dots N - 1$.
- It is guaranteed that the PIN can always be uniquely determined.
- Naturally, two entries of the PIN in the sequence do not overlap.

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 [15 points]:** The sequence contains only the PIN, without spurious digits (as in the first sample case).
- **Subtask 3 [50 points]:** $K = 4$: the PIN has *always* four digits.
- **Subtask 4 [20 points]:** $K \leq 6$.
- **Subtask 5 [15 points]:** No additional limitations.

Examples


| input.txt | output.txt |
|-----------------------------|------------|
| 8 4 1 2 3 4 1 2 3 4 | 1 2 3 4 |
| 10 4 9 2 5 0 1 2 5 0 1 1 | 2 5 0 1 |

Explanation

In the **first sample case**, the sequence candidate to be the PIN is “1 2 3 4” which appears twice.



In the **second sample case**, the sequence candidate to be the PIN is “2 5 0 1” which appears twice.



Unfair review (review)

Giorgio has just submitted his results on the advent of bloomsday to the renowned *Journal of Scientific Hoaxes*.

However, his paper has been rejected by an anonymous reviewer who claimed that Giorgio used the wrong coefficients K and E for the holy sequence!

Needless to say that Giorgio didn't take it too well, and started to stalk the journal's editors until he finally managed to get a picture of the supposedly "correct" holy sequence from the distance.

From the reviewer's report, Giorgio knows that the correct holy sequence is of the form:

$$K, 2K, 3K, 4K \dots$$



Figure 1: The editor (in red) reading the holy sequence.

Unfortunately, the pictures he managed to take are not precise enough: he was able to decipher only N numbers out of the first M of the sequence; and even of those numbers, he only deciphered their last D digits. Giorgio is sure about the relative appearance order of the digits he observed, even though he cannot tell their exact position in the sequence.

☞ The last D digits of a number X can be computed through the *modulo* operation $X \bmod 10^D$, written `X % Y` in C/C++ and `X mod Y` in Pascal (where Y is 10^D).

Given the sequence S_0, \dots, S_{N-1} of digits that Giorgio was able to decipher, help him find the smallest positive number K which could produce the observed sequence, that is, the smallest K with N increasing indexes $1 \leq a_0 < a_1 < \dots < a_{N-1} \leq M$ such that $a_i K \bmod 10^D = S_i$.

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

Input

The first line contains integers N, M, D . The second line contains N integers S_i .

Output

You need to write a single line with an integer: the smallest integer K that is compatible with the observed sequence.

Constraints

- $1 \leq D \leq 6$.
- $1 \leq N \leq M \leq 10^D$.
- $1 \leq N \leq 1000$.
- $0 \leq S_i < 10^D$ for each $i = 0 \dots N - 1$.
- It is guaranteed that a solution K 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** [20 points]: $D = 1$.
- **Subtask 3** [30 points]: $D \leq 4$.
- **Subtask 4** [25 points]: $K \leq 1000$.
- **Subtask 5** [15 points]: The final digit of K is among 1, 3, 7, 9.
- **Subtask 6** [10 points]: No additional limitations.

Examples

| input.txt | output.txt |
|-----------------------|------------|
| 3 6 1 2 8 4 | 6 |
| 3 10 1 2 8 4 | 2 |
| 4 10 2 57 95 52 90 | 19 |

Explanation

In the **first sample case**, the first six numbers of the sequence corresponding to $K = 6$ are 6, 12, 18, 24, 30, 36; which reduced to their final digit become the sequence 6, 2, 8, 4, 0, 6 which has 2, 8, 4 as a subsequence (induced by indexes $a_0 = 2, a_1 = 3, a_2 = 4$). Furthermore, smaller values of K do not produce valid sequences.

In the **second sample case**, $K = 2$ is a valid solution since 2, 8, 4 is the subsequence of 2, 4, 6, 8, 0, 2, 4, 6, 8, 0 induced by indexes $a_0 = 1, a_1 = 4, a_2 = 7$.

In the **third sample case**, $K = 19$ is a valid solution since 57, 95, 52, 90 is the subsequence of 19, 38, 57, 76, 95, 14, 33, 52, 71, 90 induced by indexes $a_0 = 3, a_1 = 5, a_2 = 8, a_3 = 10$.

Speculative execution (spectre)

Given how much Luca loves IT security, you should not be surprised to know that he wants to understand the recent flaw discovered in CPUs: Spectre!¹. The full paper describing the vulnerability is quite intricate and so he wants to proceed a step at a time.

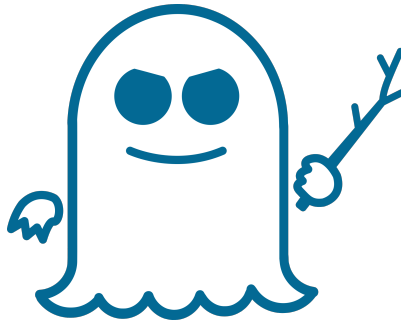


Figure 1: The official vulnerability’s logo. Cool, isn’t it?

Every respectable programmer knows that a CPU fundamentally executes instructions, one after another. But that is only one part of the story: modern CPUs aim at maximizing performance by executing multiple instructions at once, whenever possible.²

You are given a list of N instructions, each in the form

$$\text{target_variable} = \text{operand1_variable} \text{ OP } \text{operand2_variable}$$

where OP is a mathematical operator among $+$, $-$, $*$, $/$. An instruction j can be executed immediately if it does not have to “wait” that one of its operand has to be computed (more precisely, a variable needs to be computed when it is the target of another instruction i with $i < j$).

In order to let Luca check that he has understood correctly this matter, compute for him the number of instructions that can be executed immediately!

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

Input

The first line contains the only integer N , the number of instructions. The following N lines contain each a single instruction, in the format specified above.

Output

You need to write a single line with an integer: the number of instructions that can be immediately executed.

¹Spectre is probably one of the most profound vulnerabilities of this decade. More on <https://spectreattack.com/>

²The idea is called “pipelining”, some details can be read at https://en.wikipedia.org/wiki/Instruction_pipelining

Constraints

- $1 \leq N \leq 1\,000\,000$.
- Every variable name is a string consisting in 1 – 10 lower case ASCII letters.

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]**: Every variable appears only once in the whole set of instructions.
- **Subtask 3 [20 points]**: All variable names are single letters (a-z).
- **Subtask 4 [10 points]**: $N \leq 10$.
- **Subtask 5 [20 points]**: $N \leq 1000$.
- **Subtask 6 [30 points]**: No additional limitations.

Examples

| input.txt | output.txt |
|---|--------------|
| <pre>2 foo = bar + baz i = foo * i</pre> | <pre>1</pre> |
| <pre>5 width = length + deltax height = length + deltay area = width * height halfarea = area / two perimeter = length * four</pre> | <pre>3</pre> |

Explanation

In the **first sample case**, the first instruction can be executed immediately. The second instruction depends on the value of `foo`, which is computed in the first instruction; thus it *cannot* be executed at the moment.

In the **second sample case**, the first two instructions can be executed immediately. The third one depends on both the first and the second, thus cannot be executed. The fourth one depends on the third (which is on hold), thus neither can be executed. Finally, the last instruction can be executed out-of-order immediately.