

Arena Freaks (arena)

William is a long-time fanatic of *OpenArena*, an open source FPS (First-Person Shooter) game. For this reason he decided to start a brand-new web portal gathering OpenArena addicts from all over the world: **arenafreaks.net**. This website addresses multiple different needs of a professional OA player through forums, dedicated social networks, tutorials, rankings, and much more. The feature which William is most proud of, however, is the game database: a huge collection of the most spectacular games ever played in OpenArena, transcribed in *arenaic notation* (an alphanumeric notation subtly inspired by the algebraic notation of chess games).



Figure 1: A typical OpenArena “capture the flag” match.

In this notation, a game between N players (numbered from 0 to $N - 1$) each starting with L lives is transcribed as a series of E events which can be (among many others) of the following types:

- *frags* (denoted as ‘ $P \ f \ Q$ ’), where a player P “kills” a player Q thus reducing the number of Q ’s lives by 1;
- *explosions* (denoted as ‘ $P \ e$ ’), where a player P frags simultaneously all the other players, thus reducing the number of their lives by 1.

Help William count the number of players still alive (with some lives left) at the end of the game!

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

Input

The input file contains the full transcription of a game in arenaic notation. The first line contains the three integers N , E , L . Other E line follow, each containing the transcription of an event (as described in the task description).



Output

You need to write a single line with an integer: the number of players with some lives left at the end of the game.

Constraints

- $1 \leq N, E, L \leq 100\,000$.
- $0 \leq P, Q \leq N - 1$ in each event.
- All events are valid, that is, players P and Q in both frags and explosions cannot have already lost all their lives.

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 [5 points]:** Examples.
- **Subtask 2 [15 points]:** $N \leq 5$ and there are no explosions.
- **Subtask 3 [20 points]:** There are no explosions.
- **Subtask 4 [20 points]:** There are no frags.
- **Subtask 5 [20 points]:** $N, E \leq 100$.
- **Subtask 6 [20 points]:** No additional limitations.

Examples

input.txt	output.txt
4 4 2 0 f 3 1 e 2 f 1 0 f 2	2
6 7 4 5 e 5 f 1 1 f 2 0 f 2 3 f 2 5 e 3 e	4

Explanation

In the **first sample case**, the number of player's lives changes as follows:

Player	0	1	2	3
	2	2	2	2
0 f 3	2	2	2	1
1 e	1	2	1	×
2 f 1	1	1	1	
0 f 2	1	1	×	

In the **second sample case**, the number of player's lives changes as follows:

Player	0	1	2	3	4	5
	4	4	4	4	4	4
5 e	3	3	3	3	3	4
5 f 1	3	2	3	3	3	4
1 f 2	3	2	2	3	3	4
0 f 2	3	2	1	3	3	4
3 f 2	3	2	×	3	3	4
5 e	2	1		2	2	4
3 e	1	×		2	1	3

Find My Coat (coats)

During the winter William brings his coat at school, as everybody does. The only problem is that sometimes there are too many people in the classroom, and there's no space to put everybody's coat.

The classroom has a coat hanger with N hooks, each of them able to hold 1 or 2 coat (one coat on top of the other). Any more than 2 coats on a single hook would cause it to break under the excessive weight.

When students arrive, they usually try to find some empty hook to put their coat on and, if they don't find any, they will put their coat on top of another coat. If every hook has already 2 coats on it, then the student will just keep his coat (and will sweat a lot during the lecture).

Today William managed to put his coat on the rack, however, after the end of the lecture, he didn't remember on which hook he had left it.

Note that:

- The coat might be under another coat (so it might be necessary to lift the other coat in order to see it).
- William's friend Andrea provided him with information about the coat rack (he told William how many coats are on each of the N hooks).
- Since all of this is happening after the end of the lecture (and students have already started to leave the classroom) some hooks might have 0 coats, some might have 1, some 2.

Before going to look for his coat, William started wondering: "how many coats will I have to lift in order to find my coat?". William wants to lift as few coats as possible to avoid touching other people stuff (he's afraid of germs).

Since we have no information on where his coat is (it might be anywhere on the coat rack) compute the **expected number** of other people's coats that William will need to lift in order to get his coat back.

That is: if William repeated this experiment a bunch of times with the same configuration of the coat rack, how many coats would he need to lift *on average*?

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



Figure 1: This is not William.

Input

The first line contains the only integer N , the number of hooks in the rack. The second line contains N space-separated integers C_i , the number of coats on each hook.

Output

You need to write a single line with a real number: the expected number of coats to be lifted.



Constraints

- $1 \leq N \leq 100\,000$.
- $0 \leq C_i \leq 2$ 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** [5 points]: Examples.
- **Subtask 2** [25 points]: $N \leq 20$.
- **Subtask 3** [40 points]: $C_i = 2$ for all i .
- **Subtask 4** [30 points]: No additional limitations.

Examples

input.txt	output.txt
4 1 0 1 1	0
4 1 0 2 1	0.25
4 2 0 2 1	0.6

Explanation

In the **first sample case** William will immediately see his coat, thus it won't in any case be necessary to lift any coat.

In the **second sample case** it is possible that William's coat is under another coat, if it is in the third hook.

In the **third sample case** the coat can be under another coat in the first or third hook.

Note

A floating-point (real) number will be considered correct if its relative error is below 10^{-5} .

Upsetting Finals (finals)

William is getting ready for finals, at his university. This semester he has N exams available, each of them with a scheduled time T_i and an “upsetting coefficient” U_i . The upsetting value is greater for the most hated exams (some exams are so traumatic that you start to question if you really want to graduate university!).



Figure 1: William after being upset by an exam.

William will try to take as many exams as he can, but he will follow this rule: if he takes an exam at time T_i with an upsetting coefficient of U_i , then he won’t be able to endure the stress of taking any other exam in the time interval $[T_i - U_i, T_i + U_i]$.

Help William compute how many exams he can take at most.

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

Input

The first line contains the only integer N . The following N lines contain two space-separated integers T_i and U_i , the time and upsetting coefficient of the i -th exam.

Output

You need to write a single line with an integer: the max number of exams that can be taken.

Constraints

- $1 \leq N \leq 1000$.
- $1 \leq T_i \leq 1\,000\,000$ for each $i = 0 \dots N - 1$.
- $1 \leq U_i \leq 1\,000\,000$ 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** [5 points]: Examples.
- **Subtask 2** [25 points]: $N \leq 10$.
- **Subtask 3** [40 points]: Coefficients U_i are all equal.
- **Subtask 4** [30 points]: No additional limitations.

Examples

input.txt	output.txt
4 5 1 100 20 40 5 50 20	3
3 20 10 26 1 32 10	2
2 10 1 11 1	1

Explanation

In the **first sample case** William can take 3 exams: for sure he will take the one at $t = 5$ and the one at $t = 100$. The other two conflict with each other, so he will only be able to take one of them.

In the **second sample case** William can take the two most upsetting exams.

In the **third sample case** William can take either one of the exams, but not both.

Notary Club (noblesse)

Giorgio has been trying very hard to enter in the exclusive *Notary Club* during the last few years. Finally, the club administration decided to consider his application, and set a date for his admission test. Following a long-standing tradition of the Notary Club, the would-be novice now has to book (and pay for) a dinner for the whole admission committee (consisting of K grandmasters).

This is quite a problem for Giorgio, due to his very limited budget. In fact, the only place he is able to afford is “*Wu Fung’s – Typical Chinese Pizza Kebab*”, an informal take-away place which also offers N comfortable stools, the i -th of them located X_i centimeters from the start of the counter.



Figure 1: Wu Fung’s – Typical Chinese Pizza Kebab.

Giorgio knows very well *Wu Fung’s*, and is worried that the rowdy customers of the place might jeopardize his chances of success. There is only one slight chance left: to carefully book K places (not necessarily consecutive) so as to maximise the distance between the grandmasters and the rowdy customers. More precisely, given a selection of K places, we calculate the distance between the grandmasters and the rowdy customers as the *minimal distance* between a booked place and a free place. Help Giorgio calculate the maximal distance between grandmasters and customers!

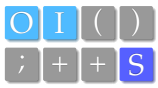
Among the attachments of this task you may find a template file `noblesse.*` with a sample incomplete implementation.

Input

The first line contains integers N , K . The second line contains N integers X_i .

Output

You need to write a single line with an integer: the maximal distance between grandmasters and customers.



Constraints

- $1 \leq K < N \leq 1000$.
- $1 \leq X_i < X_{i+1} \leq 1\,000\,000$ 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 [5 points]**: Examples.
- **Subtask 2 [15 points]**: $K = 1$.
- **Subtask 2 [20 points]**: $K = 2$.
- **Subtask 3 [30 points]**: $N \leq 20$.
- **Subtask 4 [30 points]**: No additional limitations.

Examples

input.txt	output.txt
10 2 9 10 12 25 35 40 51 64 72 80	10
6 4 10 15 30 31 32 35	15

Explanation

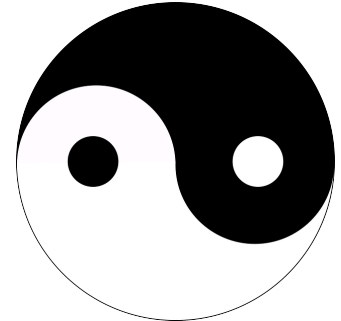
In the **first sample case**, the best option is to book places 3 and 6.

In the **second sample case**, the best option is to book all places from 2 to the end of the counter.

Array Partition (partition)

Giorgio is working on a research paper which involves the **partition** step of the Quicksort algorithm. The partition step works like this: we choose some element of the array (it doesn't matter which one), which will then be called *pivot* element. Then we move all the array elements in such a way that, in the end:


- all the elements on the left of the pivot are smaller than it;
- all the elements on the right of the pivot are greater than it.



For example, let's take this array: 2, 6, 1, 5, 7, 8, 9, 4, 3. Let's say we choose the second element as pivot (the one with value 6). On the left, there is only one element with value 2, which is smaller than 6, so the "left partitioning" is already satisfied. On the right, however, there are some elements which are *not* greater than 6, so we should move them in order to obtain a correct partition.

So, a valid partition (among many) of the array is this: 2, 3, 4, 1, 5, 6, 9, 7, 8.

For the purpose of Giorgio's research, we need to know the number of *out-of-place elements* for each possible pivot. In the example above, with pivot 6, there are 0 out-of-place elements on the left and 4 on the right (1, 5, 4 and 3), for a total of 4 out-of-place elements. If the pivot 5 was chosen, there would be $1 + 2 = 3$ out-of-place elements. Help Giorgio compute the number of out-of-place elements for each possible pivot.

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

Input

The first line contains the only integer N , the size of Giorgio's array. The second line contains N distinct integers V_i , the elements of the array.

Output

You need to write a single line with N space-separated integers, the i -th of which represents the number of out-of-place elements when the i -th pivot is chosen.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq V_i \leq N$ for each $i = 0 \dots N - 1$.
- All array elements 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** [5 points]: Examples.
- **Subtask 2** [35 points]: $N \leq 10$.
- **Subtask 3** [50 points]: $N \leq 1000$.
- **Subtask 4** [10 points]: No additional limitations.

Examples

input.txt	output.txt
9 2 6 1 5 7 8 9 4 3	1 4 2 3 2 2 2 6 6
3 3 2 1	2 2 2

Explanation

The **first sample case** is the example which was previously described.

In the **second sample case**, “every other element” is out-of-place, regardless of which pivot is chosen.

Washing Socks (socks)

Since William started attending university, he had to learn how to live on his own. He quickly mastered the art of cooking (that was an easy one, since Italian people only eat pasta) and, most importantly, how to do the laundry. William found out that he really hates washing *socks* because they come in pairs (as opposed to t-shirts and other types of clothes) so they get mixed up too easily. To avoid mixing up his socks, William found a clever solution: he will use a robot to move the socks in the “correct” place.

In the washing machine there are N socks, each of those being of a specific color specified by an integer number between 0 and $C - 1$, where C is the total number of colors. William takes each sock out of the washing machine and puts it on the clothesline. The robot will then start repeatedly swapping the sock with the adjacent one, until they match colors: when the robot finds two socks with matching colors, it will put them one on top of the other (possibly producing “stacks” of same-colored socks).



Figure 1: This is a clothesline.

For example, let's say that William takes a **blue** sock out of the washing machine when the clothesline contains the following 8 socks:

red (x2), blue, orange (x3), green, violet

In that case, after the **blue** sock is added at the rightmost end of the clothesline, the robot will swap it with the **violet** sock, then it will swap it with the **green** sock, then with the **orange** “stack” of socks, and then it will stop swapping (because the adjacent sock is of the same color) and it will create a “stack” of **blue** socks, like this:

red (x2), blue (x2), orange (x3), green, violet

The total number of swaps executed by the robot is 3. If the new sock was **red** instead, then 4 swaps would have been needed. If the new sock was **brown**, 5 swaps would have been needed.

Given the list of socks taken out of the washing machine, in their order, compute the total number of swaps that will be executed.

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

Input

The first line contains two space-separated integers N and C , respectively: the number of socks and the number of colors. The second line contains N integers S_i , the colors of the socks in the order in which they are taken from the washing machine.

Output

You need to write a single line with an integer: the total number of swaps.

Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq C \leq 1\,000\,000\,000$.
- $0 \leq S_i < C$ 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** [5 points]: Examples.
- **Subtask 2** [35 points]: $N \leq 1000$.
- **Subtask 3** [20 points]: All colors are distinct.
- **Subtask 4** [20 points]: $C \leq 1000$.
- **Subtask 5** [20 points]: No additional limitations.

Examples

input.txt	output.txt
9 5 3 1 2 4 2 0 2 0 4	21
15 2 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1	6

Explanation

The **first sample case** corresponds to the example presented in the task description, if we assume: 0=red, 1=green, 2=orange, 3=violet, 4=blue.

Tax Return (taxes)

This year Giorgio has been working on a lot of projects, each with very different characteristics. Now it is time to fill in the tax return, and it is going to be a serious mess! In particular, Giorgio has collected the following incomes:

- X euros as **Partita IVA** (independent contractor);
- Y euros as **Lavoro Occasionale** (casual work);
- Z euros as **Lavoro Dipendente** (dependent employment).

Furthermore, he has also kept a single bill of W euros due to a medical expense.



Figure 1: Giorgio hesitant to fill in the tax return.

The fiscal system in Italy is quite intricate, but for our purposes can be simplified to the following:

- **Partita IVA** income is taxed separately at 40%;
- **Lavoro Occasionale** income is tax-free up to 5000 euros, and any additional amount is taxed together with *lavoro dipendente* incomes;
- **Lavoro Dipendente** income is taxed at 20% up to 10 000 euros, and at 60% for the additional amount.

Furthermore, bills due to medical expenses can be detracted to a **single** source of income **before** computing the amount of taxes due. Splitting of a single bill between different incomes is not allowed, so Giorgio needs to pick only one of his source of incomes and then compute as described above. Help Giorgio calculate the minimum amount of taxes due, in euro cents.

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

Input

The first line contains the three integers X, Y, Z . The second line contains integer W .

Output

You need to write a single line with an integer: the minimum amount of euro cent that Giorgio is required to pay in taxes.

Constraints

- $0 \leq X, Y, Z, W \leq 1\,000\,000$.

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** [5 points]: Examples.
- **Subtask 2** [25 points]: $X = Y = 0$.
- **Subtask 3** [30 points]: $W = 0$.
- **Subtask 4** [40 points]: No additional limitations.

Examples

input.txt	output.txt
250 3400 400 500	8000
8500 9000 12000 4500	630000

Explanation

In the **first sample case**, the best option is to detract the medical expenses from the *partita IVA* income, thus paying $400 \times 20\% = 80$ euros (that is, 8 000 euro cents).

In the **second sample case**, the best option is to detract the medical expenses from the *lavoro dipendente* income, thus totalling $(9\,000 - 5\,000) + (12\,000 - 4\,500) = 11\,500$ euros to be taxed along the *lavoro dipendente* rules. The total amount of taxes is then $8\,500 \times 40\% + 10\,000 \times 20\% + 1\,500 \times 60\% = 6\,300$ euros (that is, 630 000 euro cents).