ICPC Europe Regionals

icpc global sponsor
programming tools

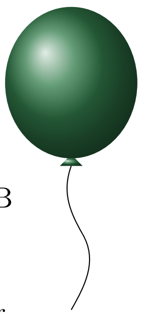
icpc.foundation

icpc diamond
multi-regional sponsor

The 2022 ICPC Southwestern Europe Regional Contest

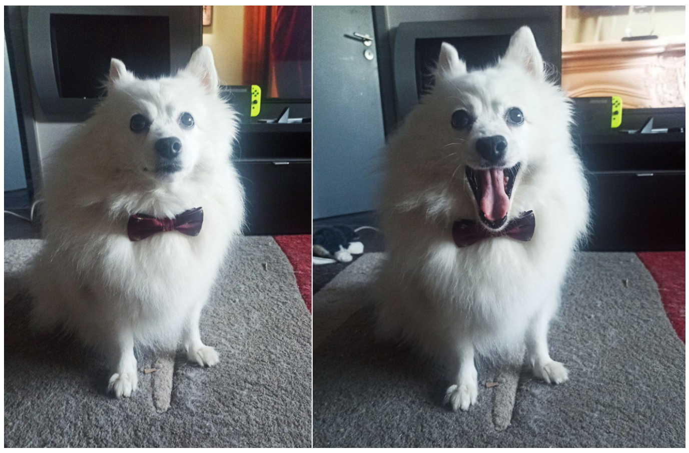
Official Problem Set

$\boxed{\text{A}}$ Walking Boy



Time limit: 2.0s
Memory limit: 2048MB

One of the SWERC judges has a dog named Boy. Besides being a good competitive programmer, Boy loves fresh air, so she wants to be walked at least twice a day. Walking Boy requires 120 **consecutive** minutes. Two walks cannot overlap, but one can start as soon as the previous one has finished.



Boy before and after getting ACCEPTED on this problem.

Today, the judge sent n messages to the SWERC Discord server. The i-th message was sent a_i minutes after midnight. You know that, when walking Boy, the judge does not send any messages, but he can send a message right before or right after a walk. Is it possible that the judge walked Boy at least twice today?

Note that a day has 1440 minutes, and a walk is considered to happen *today* if it starts at a minute $s \geq 0$ and ends right before a minute $e \leq 1440$. In that case, it must hold that $e - s = 120$ and, for every $i = 1, 2 \ldots, n$, either $a_i \leq s$ or $a_i \geq e$.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \leq t \leq 100$) — the number of test cases. The descriptions of the t test cases follow.

The first line of each test case contains an integer n ($1 \leq n \leq 100$) — the number of messages sent by the judge.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_1 < a_2 < \cdots < a_n < 1440$) — the times at which the messages have been sent (in minutes elapsed from midnight).

Output

For each test case, output one line containing `YES` if it is possible that Boy has been walked at least twice, and `NO` otherwise.

Samples

Sample input 1

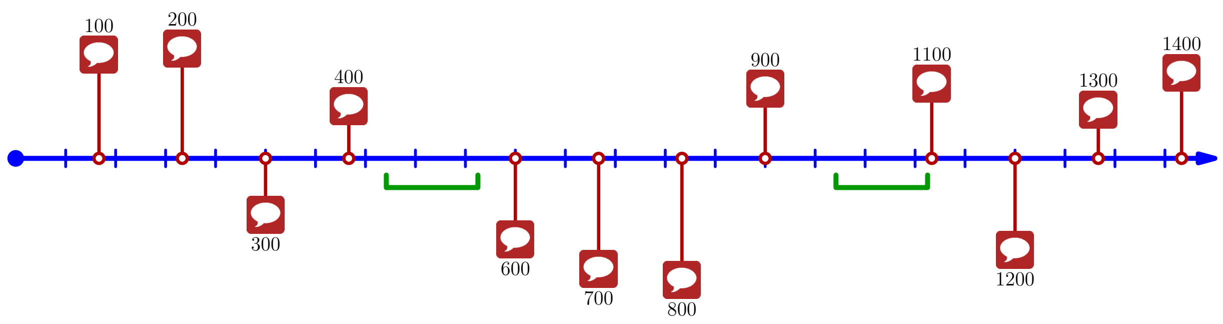
```
6
14
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
12
100 200 300 400 600 700 800 900 1100 1200 1300 1400
13
100 200 300 400 500 600 700 800 900 1100 1200 1300 1400
13
101 189 272 356 463 563 659 739 979 1071 1170 1274 1358
1
42
5
0 1 2 3 4
```

Sample output 1

```
NO
YES
NO
YES
YES
YES
```

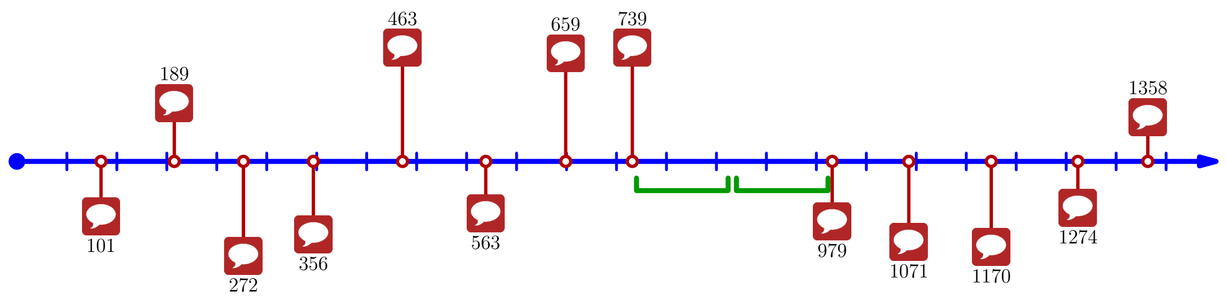
Explanation of sample 1.
In the **first test case**, the judge has sent a message at each time multiple of 100 (excluding 0). It is impossible that he has walked Boy even once.

In the **second test case**, the times are the same as above, but 500 and 1000 are missing. The judge could have walked Boy, for instance, during the time intervals $[440, 560]$ and $[980, 1100]$. The situation is illustrated in the picture below, where the walks are represented by green intervals.

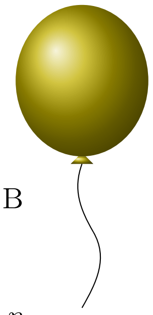


In the **third test case**, the times are the same as in the first test case, but 1000 is missing. The judge could have walked Boy at most once.

In the **fourth test case**, Boy could have been walked during the time intervals $[739, 859]$ and $[859, 979]$.



$\boxed{\text{B}}$ Uniform Chemistry



Time limit: 2.0s
Memory limit: 2048MB

In a parallel universe there are n chemical elements, numbered from 1 to n. The element number n has not been discovered so far, and its discovery would be a pinnacle of research and would bring the person who does it eternal fame and the so-called SWERC prize.

There are m independent researchers, numbered from 1 to m, that are trying to discover it. Currently, the i-th researcher has a sample of the element s_i. Every year, each researcher independently does one *fusion experiment*. In a fusion experiment, if the researcher currently has a sample of element a, they produce a sample of an element b that is chosen uniformly at random between $a+1$ and n, and they lose the sample of element a. The elements discovered by different researchers or in different years are completely independent.

The first researcher to discover element n will get the SWERC prize. If several researchers discover the element in the same year, they all get the prize. For each $i = 1, 2, \ldots, m$, you need to compute the probability that the i-th researcher wins the prize.

Input

The first line contains two integers n and m ($2 \leq n \leq 100$, $1 \leq m \leq 10$) — the number of elements and the number of researchers.

The second line contains m integers $s_1, s_2, \ldots, s_m$ ($1 \leq s_i < n$) — the elements that the researchers currently have.

Output

Print m floating-point numbers. The i-th number should be the probability that the i-th researcher wins the SWERC prize. Your answer is accepted if each number differs from the correct number by at most 10^{-8}.

Samples

Sample input 1	Sample output 1

`2 3`
`1 1 1`
	`1.0 1.0 1.0`

Explanation of sample 1.
All researchers will discover element 2 in the first year and win the SWERC prize.

Sample input 2	Sample output 2

`3 3`
`1 1 2`
	`0.5 0.5 1.0`

Explanation of sample 2.
The last researcher will definitely discover element 3 in the first year and win the SWERC prize. The first two researchers have a 50% chance of discovering element 2 and a 50% chance of discovering element 3, and only element 3 will bring them the prize.

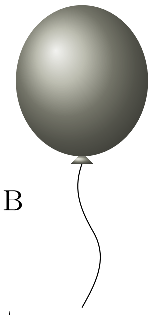
Sample input 3	Sample output 3
3 3 1 1 1	0.625 0.625 0.625

Explanation of sample 3.
Each researcher has an independent 50% chance of discovering element 3 in the first year, in which case they definitely win the SWERC prize. Additionally, if they all discover element 2 in the first year, which is a 12.5% chance, then they will all discover element 3 in the second year and all win the prize.

Sample input 4	Sample output 4

100 7
1 2 4 8 16 32 64
	0.178593469 0.179810455 0.182306771
	0.187565366 0.199300430 0.229356322
	0.348722518

$\boxed{\text{C}}$ Another Wine Tasting Event



Time limit: 0.5s
Memory limit: 2048MB

After the first successful edition, Gabriella has been asked to organize a second wine tasting event. There will be $2n-1$ bottles of wine arranged in a row, each of which is either red wine or white wine.

This time, Gabriella has already chosen the type and order of all the bottles. The types of the wines are represented by a string s of length $2n-1$. For each $1 \le i \le 2n-1$, it holds that $s_i = \texttt{R}$ if the i-th bottle is red wine, and $s_i = \texttt{W}$ if the i-th bottle is white wine.

Exactly n critics have been invited to attend. The critics are numbered from 1 to n. Just like last year, each critic j wants to taste an interval of wines, that is, the bottles at positions a_j, a_j+1, $\ldots$, b_j for some $1 \le a_j \le b_j \le 2n-1$. Moreover, they have the following additional requirements:

- each of them wants to taste at least n wines, that is, it must hold that $b_j - a_j + 1 \ge n$;
- no two critics must taste exactly the same wines, that is, if $j \ne k$ it must hold that $a_j \ne a_k$ or $b_j \ne b_k$.

Gabriella knows that, since the event is held in a coastal region of Italy, critics are especially interested in the white wines, and don't care much about the red ones. (Indeed, white wine is perfect to accompany seafood.) Thus, to ensure fairness, she would like that all critics taste the same number of white wines.

Help Gabriella find an integer x (with $0 \le x \le 2n-1$) such that there exists a valid assignment of intervals to critics where each critic tastes exactly x white wines. It can be proved that at least one such x always exists.

Input

The first line contains the integer n ($1 \le n \le 10^6$) — where $2n-1$ is the number of bottles, and n is the number of critics.

The second line contains a string s of length $2n-1$ that represents the arrangement of the wines — the i-th character of s ($1 \le i \le 2n-1$) is $\texttt{R}$ for a red wine and $\texttt{W}$ for a white wine.

Output

Print an integer x — the number of white wines that each critic will taste.

It can be proved that at least one solution exists. If multiple solutions exist, any of them will be accepted.

Samples

Sample input 1	Sample output 1

`5`
`RWWRRRWWW`
	`2`

Explanation of sample 1.
There are 5 critics and $2 \cdot 5 - 1 = 9$ bottles of wine. A possible set of intervals that makes each critic taste 2 white wines is the following: $[2, 6]$, $[1, 6]$, $[4, 8]$, $[1, 5]$, $[3, 7]$. Note that all intervals contain at least 5 bottles.

Sample input 2	Sample output 2

`1`
`R`
	`0`

Explanation of sample 2.
There is 1 critic and $2 \cdot 1 - 1 = 1$ bottle of wine. The only possible interval is $[1, 1]$, which gives $x = 0$.

$\boxed{\text{D}}$ Crossing the Railways

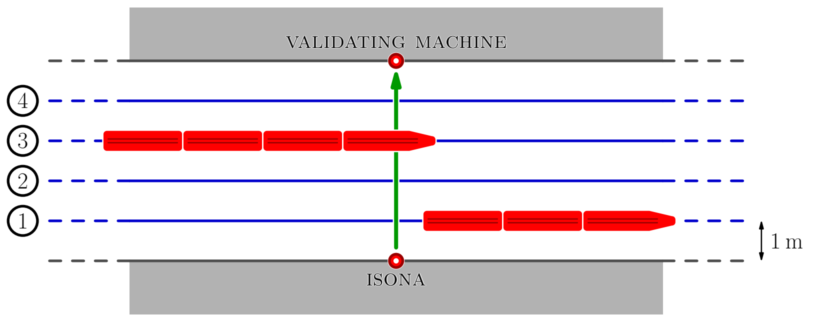
Time limit: 4.0s
Memory limit: 2048MB

Isona is in a train station. This station has two platforms, and between them there are m parallel railways that can be viewed as infinite straight lines. Each railway is identified with an integer from 1 to m, railway 1 being the closest to the first platform and railway m being the farthest. There is a 1 meter distance between consecutive railways, as well as between each platform and its closest railway.

Isona is standing on the inner border of the first platform, when she realizes that she forgot to validate her ticket! There is a validating machine on the second platform, exactly opposite her current position (thus, the distance between Isona and the validating machine is $m + 1$ meters). There are only s seconds left to validate the ticket and the bridge designated to cross the railways is too far from the validating machine. Therefore, Isona (who is very brave and a little bit careless) will cross the railways running in a straight line perpendicular to the railways themselves. Isona can only run forward (not backward) and **she can stay still**. When she runs at maximum speed, she needs v seconds to traverse 1 meter. She can run at any speed less than or equal to her maximum speed.

There is only one problem: n trains are programmed to transit through the railways. The i-th train will use the railway r_i. It will start crossing the straight line between Isona and the validating machine a_i seconds from now and it will end b_i seconds from now. Of course, Isona cannot cross a railway when a train is passing. Formally, for every $i = 1, 2, \ldots, n$, Isona is not allowed to be on railway r_i at any time t with $a_i < t < b_i$ (but she is allowed to cross at times a_i or b_i).

The following picture summarizes the situation. In the picture there are $m = 4$ railways and two trains are visible; the train going through railway 3 is currently crossing the line between Isona and the validating machine.



Isona is a really good runner, but she gets tired every time she has to change her running speed. What is the minimum number of speed changes she has to perform to get to the validating machine on the other platform within s seconds from now? Note that at the beginning Isona is not running. She can start to run anytime. The instant she starts to run (i.e. her speed becomes positive) is **not** counted as a speed change.

Input

The first line of the input contains four integers n, m, s, v ($1 \leq n \leq 500$, $1 \leq m \leq 10$, $1 \leq s, v \leq 10^9$) — the number of trains, the number of railways, the maximum time in seconds Isona can spend crossing the railways, and the number of seconds she needs to traverse 1 meter at maximum speed.

Each of the next n lines contains three integers a_i, b_i, r_i ($1 \leq a_i < b_i \leq 10^9$, $1 \leq r_i \leq m$) — the start and end times of the i-th train crossing the straight line between Isona and the validating machine, and the railway it will be using.

It is guaranteed that, for any two trains i and j that go through the same railway (i.e. $r_i = r_j$), there is at least 1 second between them (that is, either $a_j \geq b_i + 1$ or $a_i \geq b_j + 1$).

Output

Print the minimum number of speed changes Isona has to perform to get to the validating machine in time. If this is impossible, print -1.

Samples

Sample input 1	Sample output 1

4 3 5 1
1 2 1
3 4 1
2 3 2
3 4 3
	0

Explanation of sample 1.
If Isona starts running at time $t = 0$ at maximum speed (1 m/s), she will cross each railway just when a train is about to traverse it, and she will arrive at the other platform at time $4 = s - 1$ without changing speed.

Sample input 2	Sample output 2

3 3 12 2
2 10 1
1 6 2
8 12 3
	2

Explanation of sample 2.
A possible solution with 2 speed changes is the following: for the first 2 seconds Isona goes at maximum speed (0.5 m/s), then she slows down to 0.25 m/s for 4 seconds until she reaches the second railway. At that point, she goes at maximum speed again until she reaches the other platform.

Sample input 3	Sample output 3

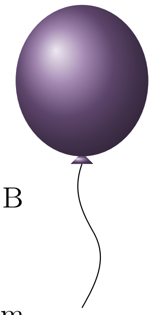
`8 4 13 2`
`1 4 1`
`5 13 1`
`1 5 2`
`6 13 2`
`1 9 3`
`10 13 3`
`1 10 4`
`11 13 4`
	`2`

Explanation of sample 3.
Isona can wait 2 seconds before starting running. She then runs for 5 seconds at maximum speed (0.5 m/s). After that, she waits for 1 second not running (or running at 0 m/s), and finally she runs again at maximum speed for the last 5 seconds. Overall, she changes speed twice.

Sample input 4	Sample output 4

`1 1 2 2`
`1 2 1`
	`-1`

E Spinach Pizza



TIME LIMIT: 2.0S
MEMORY LIMIT: 2048MB

The two siblings Alberto and Beatrice have to eat a spinach pizza together. However, none of them likes spinach, so they both want to eat as little as possible.

The pizza has the shape of a strictly convex polygon with n vertices located at integer coordinates (x_1, y_1), (x_2, y_2), $\dots$, (x_n, y_n) of the plane.

The siblings have decided to eat the pizza in the following way: taking turns, starting with Alberto, each sibling chooses a vertex of the remaining part of the pizza and eats out the triangle determined by its two neighboring edges. In this way, after each of the first $n-3$ turns the pizza will have one less vertex. The game ends after the $(n-2)$-th turn, when all the pizza has been eaten.

Assuming that Alberto and Beatrice choose the slices to eat optimally, which of the siblings manages to eat **at most half** of the pizza? You should identify a sibling that has a strategy to do so and help them choose the slices appropriately. Note that it is possible that both Alberto and Beatrice end up eating exactly half of the area if they choose their slices optimally.

INPUT

The first line contains a single integer n ($4 \le n \le 100$) — the number of vertices.

The next n lines contain two integers x_i and y_i each ($-10^6 \le x_i, y_i \le 10^6$) — the coordinates of the i-th vertex of the polygon representing the initial shape of the pizza.

It is guaranteed that the polygon is strictly convex and that its vertices are given in counterclockwise order.

INTERACTION

First, you should print a line containing either the string `Alberto` or the string `Beatrice` — the sibling that you will help to win.

Then, for the next $n-2$ turns, you will alternate with the judge in choosing a slice of the pizza and removing it, starting with you if you chose to help Alberto, or starting with the judge if you chose to help Beatrice.

- When it is your turn, print a single line containing an integer p ($1 \le p \le n$) that has not been chosen before, indicating that you want to eat the slice determined by the vertex located at (x_p, y_p).
- When it is the judge's turn, read an integer q ($1 \le q \le n$), indicating that the other player eats the slice determined by the vertex located at (x_q, y_q). It is guaranteed that q has not been chosen before.

If one of your interactions is malformed, the interactor terminates immediately and your program receives the verdict `WRONG-ANSWER`. Otherwise, you will receive `CORRECT` if at the end your player has eaten at most half of the pizza, and `WRONG-ANSWER` otherwise.

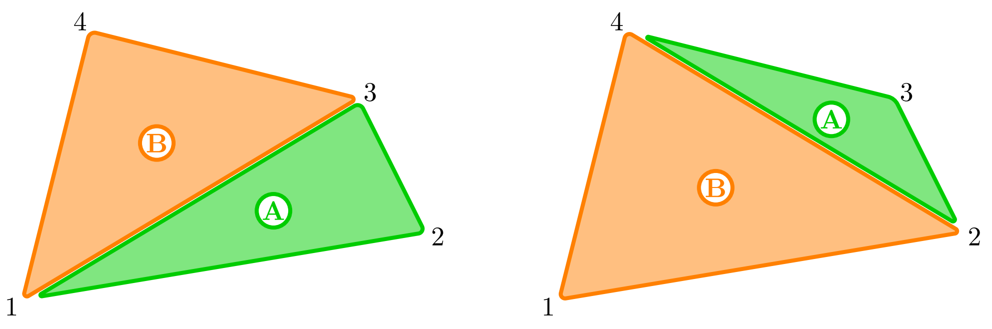
After printing a line do not forget to **end the line** and **flush the output**. Otherwise, you will get the verdict `TIMELIMIT`. To flush the output, use:

- `fflush(stdout)` in C;
- `fflush(stdout)`, `cout << flush` or `cout.flush()` in C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.

Samples

Sample input 1	Sample output 1
`4`	`-`
`0 0`	
`6 1`	
`5 3`	
`1 4`	

Explanation of sample 1.
The pizza has area 15. Alberto can eat less than half of the pizza by eating the slice around vertex 2 (which has area 6.5) or around vertex 3 (which has area 3.5).



Sample input 2	Sample output 2
`6`	`-`
`0 0`	
`2 0`	
`3 2`	
`2 4`	
`0 4`	
`-1 2`	

Explanation of sample 2.
It can be proved that both players will eat exactly half of the pizza if they eat optimally. Therefore it is possible to choose to help either Alberto or Beatrice.

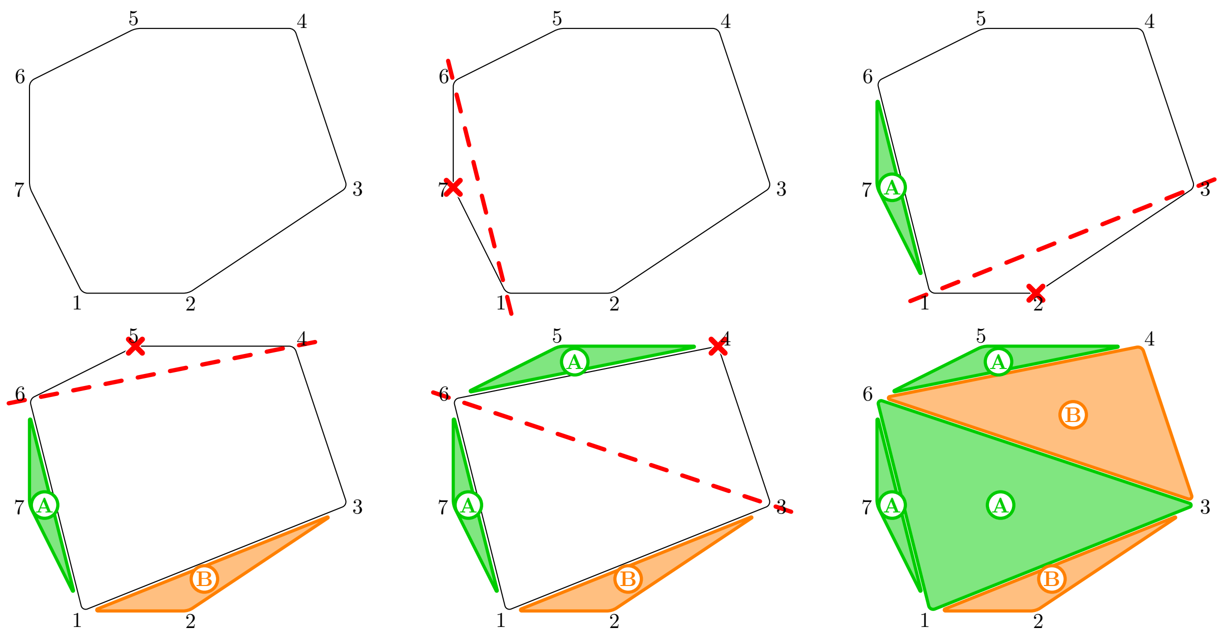
Sample input 3	Sample output 3

7
0 0
2 0
5 2
4 5
1 5
-1 4
-1 2
	-

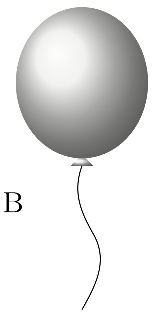
Explanation of sample 3.
It is possible to show that only Beatrice has a strategy to eat at most half of the pizza. The following is an example of a valid interaction (after reading the input):

Contestant	Judge	Explanation
`Beatrice`		The contestant will help Beatrice
	7	Alberto eats the triangle with vertices 6, 7, 1 and area 1
2		Beatrice eats the triangle with vertices 1, 2, 3 and area 2
	5	Alberto eats the triangle with vertices 4, 5, 6 and area 1.5
4		Beatrice eats the triangle with vertices 3, 4, 6 and area 8
	6	Alberto eats the triangle with vertices 3, 6, 1 and area 11

The total area eaten by Alberto is 13.5 and the total area eaten by Beatrice is 10, which is less than half the area of the whole pizza. The actions performed by the contestant and the judge in this example of interaction may be non-optimal. The process is illustrated below:



$\boxed{\text{F}}$ Beppa and SwerChat



Time limit: 2.0s
Memory limit: 2048MB

Beppa and her circle of geek friends keep up to date on a group chat in the instant messaging app SwerChat$^{\text{TM}}$.

The group has n members, excluding Beppa. Each of those members has a unique ID between 1 and n. When a user opens a group chat, SwerChat$^{\text{TM}}$ displays the list of other members of that group, sorted by **decreasing** times of last seen online (so the member who opened the chat most recently is the first of the list). However, the times of last seen are not displayed.

Today, Beppa has been busy all day: she has only opened the group chat twice, once at 9:00 and once at 22:00. Both times, she wrote down the list of members in the order they appeared at that time. Now she wonders: what is the minimum number of other members that must have been online at least once between 9:00 and 22:00?

Beppa is sure that no two members are ever online at the same time and no members are online when Beppa opens the group chat at 9:00 and 22:00.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 10\,000$) — the number of test cases. The descriptions of the t test cases follow.

The first line of each test case contains an integer n ($1 \le n \le 10^5$) — the number of members of the group excluding Beppa.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the list of IDs of the members, sorted by decreasing times of last seen online at 9:00.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$) — the list of IDs of the members, sorted by decreasing times of last seen online at 22:00.

For all $1 \le i < j \le n$, it is guaranteed that $a_i \neq a_j$ and $b_i \neq b_j$.

It is also guaranteed that the sum of the values of n over all test cases does not exceed 10^5.

Output

For each test case, print the minimum number of members that must have been online between 9:00 and 22:00.

Samples

Sample input 1	Sample output 1

4
5
1 4 2 5 3
4 5 1 2 3
6
1 2 3 4 5 6
1 2 3 4 5 6
8
8 2 4 7 1 6 5 3
5 6 1 4 8 2 7 3
1
1
1
	2
	0
	4
	0

Explanation of sample 1.
In the **first test case**, members $4, 5$ must have been online between 9:00 and 22:00.

In the **second test case**, it is possible that nobody has been online between 9:00 and 22:00.

G Parmigiana With Seafood

Time limit: 2.0s
Memory limit: 2048MB

The “Parmigiana di melanzane” is a typical Italian dish. Alessandro and Bianca have very different tastes when it comes to it: Alessandro loves to eat Parmigiana with seafood, but Bianca thinks it is an atrocity! To decide which ingredients to include in the dish they prepare, they play the following game.

There are n possible ingredients, labeled from 1 to n. The higher the label, the closer the ingredient is to being seafood. The ingredients are connected by $n-1$ edges, in such a way as to form a tree. Alessandro and Bianca take turns, with Alessandro going first. They alternately choose a *terminal ingredient* x, that is an ingredient currently connected to at most one other ingredient, and remove it from the tree. If the terminal ingredient x was chosen by Alessandro, it goes in the recipe; if it was chosen by Bianca, it is discarded.

The *taste* of the Parmigiana is measured as the maximum label of an ingredient in the recipe. Alessandro wants to maximize the taste, while Bianca wants to minimize the taste. If both play optimally, what is the taste of the Parmigiana?

Input

The first line contains an integer n ($2 \leq n \leq 100\,000$) — the number of ingredients.

Each of the following $n-1$ lines contain two integers u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) — the ingredients that the i-th edge connects.

It is guaranteed that the edges form a tree (i.e., any pair of ingredients is connected by the edges, possibly indirectly).

Output

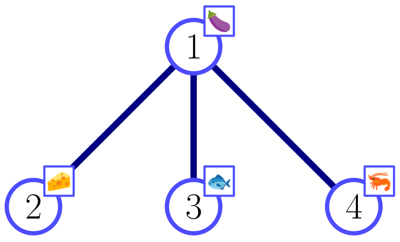
Print the value of the taste if both Alessandro and Bianca play optimally.

Samples

Sample input 1	Sample output 1

4
1 2
1 3
1 4
	4

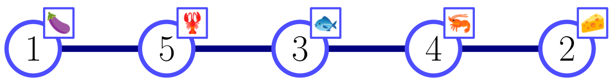
Explanation of sample 1.
Alessandro can choose terminal ingredient 4 in the first turn. This ingredient is added to the recipe. Since 4 is the maximum label of an ingredient, the taste is 4 regardless of the choices that follow.



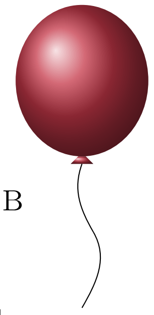
Sample input 2	Sample output 2

5
1 5
5 3
3 4
4 2
	3

Explanation of sample 2.
Bianca can make sure that neither ingredient 4 nor 5 are included in the recipe, in which case Alessandro can include 3. Thus, the taste is 3.



H Controllers



Time limit: 2.0s
Memory limit: 2048MB

You are at your grandparents' house and you are playing an old video game on a strange console. Your controller has only two buttons and each button has a number written on it.

Initially, your score is 0. The game is composed of n rounds. For each $1 \leq i \leq n$, the i-th round works as follows.

On the screen, a symbol s_i appears, which is either `+` (*plus*) or `-` (*minus*). Then you must press one of the two buttons on the controller **once**. Suppose you press a button with the number x written on it: your score will increase by x if the symbol was `+` and will decrease by x if the symbol was `-`. After you press the button, the round ends.

After you have played all n rounds, you win if your score is 0.

Over the years, your grandparents bought many different controllers, so you have q of them. The two buttons on the j-th controller have the numbers a_j and b_j written on them. For each controller, you must compute whether you can win the game playing with that controller.

Input

The first line contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of rounds.

The second line contains a string s of length n — where s_i is the symbol that will appear on the screen in the i-th round. It is guaranteed that s contains only the characters `+` and `-`.

The third line contains an integer q ($1 \leq q \leq 10^5$) — the number of controllers.

The following q lines contain two integers a_j and b_j each ($1 \leq a_j, b_j \leq 10^9$) — the numbers on the buttons of controller j.

Output

Output q lines. On line j print `YES` if the game is winnable using controller j, otherwise print `NO`.

Samples

Sample input 1	Sample output 1

```
8
+-+---+-
5
2 1
10 3
7 9
10 10
5 3
```
	```
	YES
	NO
	NO
	NO
	YES
	```

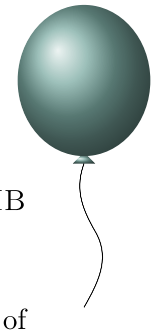
Explanation of sample 1.
One possible way to get score 0 using the first controller is by pressing the button with numnber 1 in rounds 1, 2, 4, 5, 6 and 8, and pressing the button with number 2 in rounds 3 and 7. It is possible to show that there is no way to get a score of 0 using the second controller.

Sample input 2	Sample output 2

`6`
`+-++--`
`2`
`9 7`
`1 1`
	`YES`
	`YES`

Sample input 3	Sample output 3

`20`
`+-----+--+--------+-`
`2`
`1000000000 99999997`
`250000000 1000000000`
	`NO`
	`YES`



$\boxed{\text{I}}$ Library game

TIME LIMIT: 2.0S
MEMORY LIMIT: 2048MB

Alessia and Bernardo are discovering the world of competitive programming through the books of their university library.

The library consists of m sections numbered from 1 to m. Each section contains only books dedicated to a particular subject and different sections correspond to different subjects. In order to prevent the students from wandering in the library, the university has established a system of passes. Each pass has a length y associated to it and allows access to an interval of y consecutive sections in the library. During a visit, the student must choose exactly one book from one of these sections and leave the library. Each pass can be used only once.

At the moment Alessia and Bernardo have n passes of lengths $x_1, x_2, \ldots, x_n$. They have different opinions on the best way to improve: Alessia thinks that it is important to study many different topics, while Bernardo believes that it is important to study deeply at least one topic. So, Alessia wants to use the n passes to get n books on distinct topics, while Bernardo would like to get at least two books on the same topic.

They have reached the following agreement: for each of the following n days, Alessia will choose a pass of length y among those which are still available and an interval of y sections in the library, and Bernardo will go into the library and will take exactly one book from one of those sections.

Can Bernardo manage to get at least two books on the same subject, or will Alessia be able to avoid it?

You should decide whether you want to be Alessia or Bernardo, and you have to fulfill the goal of your chosen character. The judge will impersonate the other character. Note that, even if at some moment Bernardo has already taken two books on the same subject, the interaction should go on until the end of the n days.

Input

The first line contains two integers n and m ($1 \leq n \leq 100$, $n \leq m \leq 5000$) — the number of passes and the number of sections.

The second line contains n integers $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq m$) — the lengths of the passes available.

Interaction

First, you should print a line containing either the string `Alessia` or the string `Bernardo` — the character that you want to impersonate.

Then, for each of the n turns:

- If you are taking the role of Alessia, print a single line consisting of two integers y and a ($1 \leq a \leq m - y + 1$) — you are choosing a pass of length y and the interval of sections $[a, a + y - 1]$. Note that at least one pass of length y must still be available. After this, read a single integer b ($a \leq b \leq a + y - 1$) — the subject of the book selected by Bernardo.

- If you are taking the role of Bernardo, read two integers y and a ($1 \leq a \leq m - y + 1$) — the length of the pass chosen by Alessia and the index of the first section of the interval. It is guaranteed that there is at least one pass of length y available. Then, print a line consisting of a single integer b ($a \leq b \leq a + y - 1$) — the subject from which you choose to take a book.

If one of your interactions is malformed, the interactor terminates immediately and your program receives the verdict `WRONG-ANSWER`. Otherwise, you will receive the verdict according to the game's criteria described above.

After printing a line do not forget to **end the line** and **flush the output**. Otherwise, you will get the verdict `TIMELIMIT`. To flush the output, use:

- `fflush(stdout)` in C;
- `fflush(stdout)`, `cout << flush` or `cout.flush()` in C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python.

Samples

Sample input 1	Sample output 1

`5 14`
`3 7 2 3 10`
	`-`

Explanation of sample 1.
It can be shown that Alessia can accomplish her goal. An example of interaction (after reading the input) follows:

Contestant	Judge	Explanation
`Alessia`		The program will act as Alessia
3 11		Choose $y = 3$ and $a = 11$
	13	Judge selects $b = 13$
10 2		Choose $y = 10$ and $a = 2$
	9	Judge selects $b = 9$
7 1		Choose $y = 7$ and $a = 1$
	4	Judge selects $b = 4$
2 10		Choose $y = 2$ and $a = 10$
	10	Judge selects $b = 10$
3 6		Choose $y = 3$ and $a = 6$
	7	Judge selects $b = 7$

The program of the contestant wins because all the books chosen by Bernardo pertain to different topics. The actions performed by the contestant and the judge in this example of interaction may be non-optimal.

Sample input 2	Sample output 2

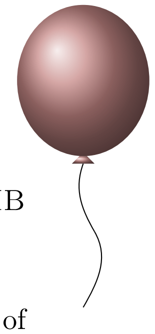
`4 10`
`4 1 6 4`
	`-`

Explanation of sample 2.
It can be shown that Bernardo can manage to fulfil his goal. An example of interaction (after reading the input) follows:

Contestant	Judge	Explanation
`Bernardo`		The program will act as Bernardo
	4 1	Judge chooses $y = 4$ and $a = 1$
4		Select $b = 4$
	1 10	Judge chooses $y = 1$ and $a = 10$
10		Select $b = 10$
	6 3	Judge chooses $y = 6$ and $a = 3$
4		Select $b = 4$
	4 5	Judge chooses $y = 4$ and $a = 5$
8		Select $b = 8$

The program of the contestant wins because Bernardo has selected two books on topic number 4. The actions performed by the contestant and the judge in this example of interaction may be non-optimal.

J Italian Data Centers



Time limit: 2.0s
Memory limit: 2048MB

An *Italian data center* consists of a set of servers, each colored green, white, or red, and a set of wires connecting them. Each wire connects two distinct servers and two servers are connected by at most one wire. Additionally, the data center is connected, i.e. there is a way to transmit information between any two servers through a sequence of wires.

To judge all of the contestant submissions, SWERC has an Italian data center. Since every year the number of contestants doubles, the data center needs to grow to adapt to the extra load. To address this, SWERC builds a new data center based on the previous year's one by following these steps:

- For each server s in the old data center, the new data center contains two servers s_1 and s_2 of the same color as s. A wire is placed connecting the two servers s_1 and s_2.
- For each wire connecting servers s and t in the old data center: if s and t have the same color, then a wire is placed in the new data center connecting s_1 and t_1 and another wire connecting s_2 and t_2; otherwise, a wire is placed in the new data center connecting s_1 and t_2 and another one connecting s_2 and t_1.

One can show that if the old data center is connected than the new data center is also connected.

You are given the Italian data center that SWERC currently has, which contains n servers (indexed by 1, 2, …, n) and m wires connecting them. The organization wants to know how good their data center will be after k years, so you should determine the *diameter* of the data center SWERC will have in k years. The diameter of the data center is the largest distance between any two servers, i.e. the shortest number of wires that have to be used to transmit something between the two servers.

Input

The first line contains three integers n, m and k ($2 \le n \le 100$, $n-1 \le m \le n(n-1)/2$, $0 \le k \le 100$) — the number of servers, the number of wires, and the number of years to consider.

The second line contains n integers c_1, c_2, …, c_n ($1 \le c_i \le 3$) — c_i is the color of server i (where 1 stands for *green*, 2 for *white* and 3 for *red*).

Each of the next m lines contains two integers s_i and t_i ($1 \le s_i, t_i \le n$) — the two servers the i-th wire connects.

It is guaranteed that the data center is connected, the endpoints of each wire are distinct, and that there are no repeated wires.

Output

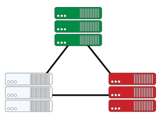
Print the diameter of SWERC's data center after k years.

Samples

Sample input 1	Sample output 1

3 3 0
1 2 3
1 2
2 3
3 1
	1

Explanation of sample 1.
The Italian data center is the following:



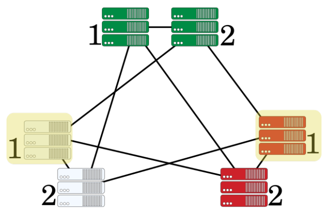
The distance between any pair of servers is 1 so the diameter is 1.

Sample input 2	Sample output 2

3 3 1
1 2 3
1 2
2 3
3 1
	2

Explanation of sample 2.
The initial Italian data center is the one from the first sample.

After one year we obtain the following (where the numbers indicate which copy the server refers to):

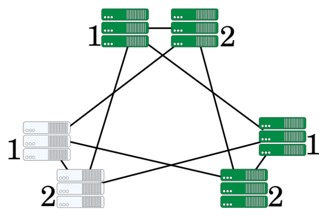


Consider the highlighted servers. The distance between them is 2 and there is no pair of servers with greater distance, so the diameter is 2.

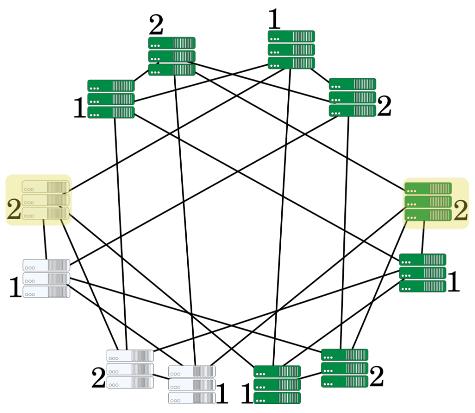
Sample input 3	Sample output 3

3 3 2
1 2 1
1 2
2 3
3 1
	3

Explanation of sample 3.
The data center after one year is the following:



After one more year:

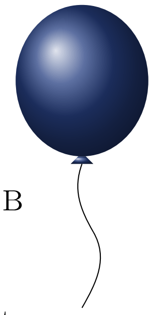


Consider the highlighted servers. The distance between them is 3 and there is no pair of servers with greater distance, so the diameter is 3.

Sample input 4	Sample output 4

`8 14 100`
`3 3 2 2 1 2 2 1`
`2 7`
`1 5`
`7 8`
`4 6`
`2 8`
`1 8`
`2 6`
`6 7`
`1 6`
`1 4`
`3 5`
`1 3`
`4 5`
`5 7`
	`53`

K Train Splitting



Time limit: 2.0s
Memory limit: 2048MB

There are n big cities in Italy, and there are m train routes between pairs of cities. Each route connects two different cities bidirectionally. Moreover, using the trains one can reach every city starting from any other city.

Right now, all the routes are operated by the government-owned Italian Carriage Passenger Company, but the government wants to privatize the routes. The government does not want to give too much power to a single company, but it also does not want to make people buy a lot of different subscriptions. Also, it would like to give a fair chance to all companies. In order to formalize all these wishes, the following model was proposed.

There will be $k \geq 2$ private companies indexed by $1, 2, \ldots, k$. Each train route will be operated by exactly one of the k companies. Then:

- For any company, there should exist two cities such that it is impossible to reach one from the other using only routes operated by that company.
- On the other hand, for any two companies, it should be possible to reach every city from any other city using only routes operated by these two companies.

Find a plan satisfying all these criteria. It can be shown that a viable plan always exists. Please note that you can choose the number k and you do not have to minimize or maximize it.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \leq t \leq 1000$) — the number of test cases. The descriptions of the t test cases follow.

The first line of each test case contains two integers n and m ($3 \leq n \leq 50$, $n-1 \leq m \leq n(n-1)/2$) — the number of cities and the number of train routes.

The next m lines contain two integers u_i and v_i each ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) — the i-th train route connects cities u_i and v_i.

It is guaranteed that the routes connect m distinct pairs of cities. It is guaranteed that using the trains one can reach every city starting from any other city.

The sum of the values of n over all test cases does not exceed 5000.

Output

For each test case, on the first line print an integer k ($2 \leq k \leq m$) — the number of companies in your plan; on the second line print m integers $c_1, c_2, \ldots, c_m$ ($1 \leq c_i \leq k$) — in your plan company c_i operates the i-th route.

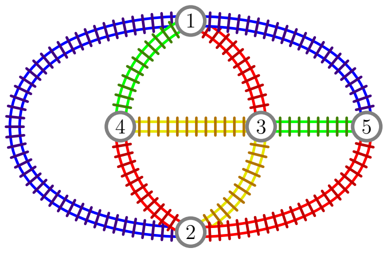
If there are multiple valid plans, you may print any of them.

Samples

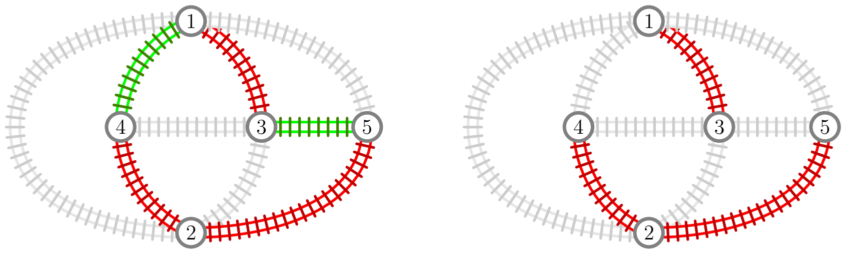
Sample input 1	Sample output 1

```
2
5 9
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
3 3
1 2
3 1
2 3
```
	```
	4
	1 2 3 1 4 2 2 4 3
	3
	2 3 1
	```

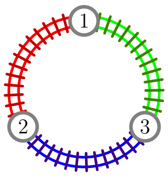
Explanation of sample 1.
In the **first test case**, the output is illustrated in the following picture, where different colors correspond to different companies (blue for 1, red for 2, green for 3, and yellow for 4):



If we consider, for example, only companies 2 and 3, we can see that from any city it is possible to reach every other city (picture on the left below). However, if we restrict to company 2 alone, it becomes impossible to reach city 5 from city 1 (picture on the right).



In the **second test case**, the output is illustrated in the following picture:



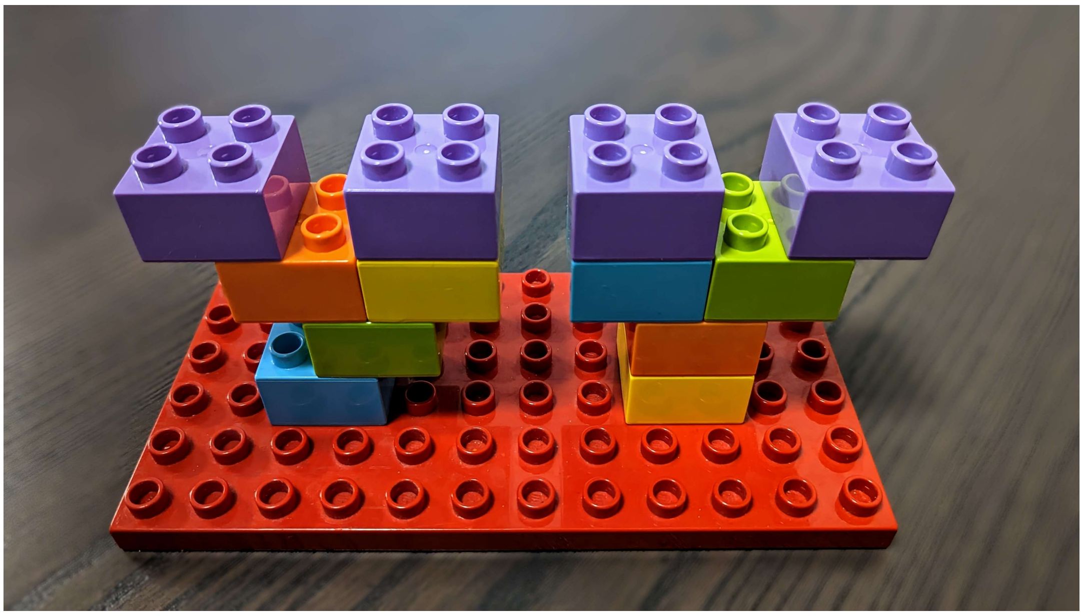
L Vittorio Plays with LEGO Bricks

Time limit: 2.0s
Memory limit: 2048MB

Vittorio is playing with his new LEGO Duplo bricks. All the bricks have the shape of a square cuboid with a 2×2 square base and a height of 1. They can be arranged in the 3D space to build structures, provided that the following rules are met:

1. No two bricks can intersect, but they can touch on their faces.
2. The corners of every brick must have integer coordinates (so bricks are axis-aligned) and the z coordinates of all corners must be non-negative.
3. The square bases of every brick must be parallel to the ground (i.e. the plane $z = 0$).
4. The lower base of any brick that is not touching the ground must touch the upper base of some other brick in a region of positive area (when this happens, the two bricks stay attached to each other thanks to small studs).

For example, this is a valid structure:



Vittorio wants to build a structure that includes purple bricks in the following n positions: $(x_1, 0, h)$, $(x_2, 0, h)$, ..., $(x_n, 0, h)$ — these are the coordinates of the centers of their lower bases; note that all of these bricks have y coordinate equal to 0 and z coordinate equal to h. Vittorio will use additional bricks of other colors to support the purple bricks. He is willing to place bricks only in positions where the center of the lower base has y coordinate equal to 0. What is the minimum number of additional bricks needed?

It can be shown that a valid construction always exists.

Input

The first line contains two integers n and h ($1 \leq n \leq 300$, $0 \leq h \leq 10^9$) — the number of purple bricks and their common z coordinate.

The second line contains n integers $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq 10^9$, $x_i + 1 < x_{i+1}$) — the x coordinates of the purple bricks (centers of the bases), given in increasing order.

Output

Print the minimum number of additional bricks needed.

Samples

Sample input 1	Sample output 1
4 0 2 7 11 13	0

Explanation of sample 1.
All the purple bricks lie on the ground, so no additional bricks are needed.

Sample input 2	Sample output 2
4 1 2 7 11 13	3

Explanation of sample 2.
Vittorio will have to place supporting bricks under the purple bricks, and he can use a single brick to support both the third and the fourth purple bricks. For example, he can place additional bricks at positions $(3, 0, 0)$, $(7, 0, 0)$ and $(12, 0, 0)$. It can be shown that it is impossible to build a valid construction using less than 3 additional bricks.

Sample input 3	Sample output 3
4 100 2 7 11 13	107

Sample input 4	Sample output 4
4 3 2 5 8 11	8

Explanation of sample 4.
A possible structure that minimizes the number of additional bricks is shown in the problem description.