

## Treatment Planning (antibiotics)

William just got an aggressive form of the seasonal flu, and the doctor prescribed him a broad-spectrum antibiotic to fight it, the *Panaceaxyl*<sup>TM</sup>. This antibiotic has to be assumed in a strictly controlled pattern, exactly once every  $T$  hours, until the flu is defeated: never stop a treatment early, or the illness will get much worse! Even the slightest transgression to the prescription, such as anticipating the timing by just a few minutes, might result in one of the numerous *Panaceaxyl*<sup>TM</sup> side effects.



Figure 1: List of *Panaceaxyl*<sup>TM</sup> side effects, as reported in 0.1 sec by the drug's commercial.

However, William is a very busy person, having already taken  $N$  commitments on his agenda, each on a different hour  $H_i$ . It would not be a good idea to plan the antibiotic for one of these hours: William is definitely going to forget about it if he is distracted by other business! More precisely, if William is going to start the treatment in hour  $X$ , he needs to have hours:

$$X, X + T, X + 2T, \dots, X + iT, \dots$$

free from commitments for all  $i > 0$ . Help William calculate the first hour in which he can start the treatment without risking the dreadful side effects.

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

### Input

The first line contains integers  $N, T$ . The second line contains  $N$  integers  $H_i$ .

### Output

You need to write a single line with an integer: the smallest hour  $X$  that is safe for starting the treatment.



## Constraints

- $1 \leq N \leq 100\,000$ .
- $1 \leq T \leq 10^9$ .
- $0 \leq H_i \leq 10^9$  for each  $i = 0 \dots N - 1$ .
- There are no repeated commitments.

## 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** [10 points]:  $T = 1$ .
- **Subtask 3** [30 points]:  $N, T, H_i \leq 1000$  for all  $i$ .
- **Subtask 4** [25 points]:  $N, T \leq 1000$ .
- **Subtask 5** [20 points]:  $T \leq 1000$ .
- **Subtask 6** [10 points]: No additional limitations.

## Examples

input.txt	output.txt
6 8 16 1 4 10 15 12	3
9 3 16 9 20 11 13 3 0 12 2	15

## Explanation

In the **first sample case**, starting hour 0 is not feasible due to the commitment in hour 16, hour 1 is already occupied and hour 2 is not feasible due to the commitment in hour 10. Starting hours 5 and 6 are also feasible but not optimal.

In the **second sample case**, any hour before 15 is not feasible due to one of the commitments in hours 12, 16 or 20.

## Capture the Flag (arena2)

William's new web portal [arenafreaks.net](http://arenafreaks.net) is struggling to take off. As the large bill from the virtual machines provider is expected to arrive sooner and sooner, William is getting frenzied by the addition of new features to the portal, and in particular to its flagship: the game database. As a matter of fact, a couple of days ago William added a database section for *capture-the-flag* games, as usual transcribed in *arenaic notation*.



Figure 1: The earliest known reportage of a *capture-the-flag* game (E. Delacroix, 1830).

The transcription of a capture-the-flag game between  $N$  teams in arenaic notation consists in a series of  $E$  events, each described by two integers  $X_i$  and  $Y_i$  from 0 to  $N - 1$  representing:

- if  $X_i \neq Y_i$ , that team  $X_i$  stole  $Y_i$ 's flag and is running to score a point;
- if  $X_i = Y_i$ , that team  $X_i$  recovered his flag (from the last team which stole it) preventing them from scoring a point.

Help William calculate the final score of each game given its transcription in arenaic notation, assuming that each point that is not later prevented is eventually scored.

Among the attachments of this task you may find a template file `arena2.*` with a sample incomplete implementation.

### Input

The first line contains the two integers  $N$  and  $E$ . The following  $E$  lines contains integers  $X_i, Y_i$ .

### Output

You need to write a single line with  $N$  integer: the score of each team in the match.

## Constraints

- $2 \leq N \leq 10\,000$ .
- $1 \leq E \leq 1\,000\,000$ .
- $0 \leq X_i, Y_i \leq N - 1$  for each  $i = 0 \dots N - 1$ .
- The events are consistent, that is, there is no “flag recover” event without a “flag conquer” event about the same flag preceding it (and after the previous “flag recover” event).

## 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]:**  $N = 2$ .
- **Subtask 3 [40 points]:**  $N \leq 10$ .
- **Subtask 4 [30 points]:** No additional limitations.

## Examples

input.txt	output.txt
<pre>3 4 0 2 1 2 2 0 2 2</pre>	<pre>1 0 1</pre>
<pre>2 7 0 1 1 0 1 1 0 0 1 0 0 1 0 1</pre>	<pre>2 1</pre>

## Explanation

In the **first sample case**, the second event (team 1 captures flag 2) is prevented by the last event (team 2 recovers flag 2), thus only teams 0 and 2 score a point.

In the **second sample case**, the first event is prevented by the third and the second event by the fourth. The last three events determine the final 2 – 1 score.

## Boring Chores (chores)

The best thing about *online* programming contests is that you don't need to set up an actual server, you just connect to your *online server* and then install what you need. For *offline* contests, instead, you are often faced with the hard reality: servers break, burst into flames, and you always need to replace parts.

Having prepared a lot of offline contests, William and Giorgio know full well how boring these “chores” can be, so they are now trying to predict how much time will be lost to them. The main problem, is that you don't know in advance where you will need to go: it may be possible that the parts you need are located in *some far away city*, and retrieving them may take a long time.

Furthermore, the contest venue may be decided at the last minute, so we don't even know in advance *which city will host the contest!*

Luckily, Giorgio has a map of all the cities and a selected set of roads (the “best” ones), so that there is exactly one path between any two cities following the map. The roads are bidirectional.

The contest plan (after the contest city is revealed) is this:

- William will check which parts of the server need to be replaced.
- Then he will go to take the replacement parts.
- Finally, he will come back to the contest city.

Given that we don't know which city will host the contest, nor which city has the server parts we need, help William compute how long it will take him *in the worst case* to go take the server parts and then come back to the contest city.

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



Figure 1: Doing chores can be very tedious.

### Input

The first line contains the only integer  $N$ . The next  $N - 1$  lines contain pairs of integers  $A_i$  and  $B_i$ . Each pair represents a bidirectional road between the  $A_i$ -th and  $B_i$ -th cities.

### Output

You need to write a single line with an integer: the total “number of roads” it will take from the contest venue to the farthest city and come back home (in the worst case).



## Constraints

- $1 \leq N \leq 100\,000$ .
- $0 \leq A_i, B_i \leq N - 1$  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** [20 points]: The cities lay on a single line of roads.<sup>1</sup>
- **Subtask 3** [20 points]:  $N \leq 10$ .
- **Subtask 4** [25 points]:  $N \leq 100$ .
- **Subtask 5** [30 points]: No additional limitations.

## Examples

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

## Explanation

In the **first sample case** it could happen that the contest venue is on the 3-th city and the parts are in the 2-th city. In this case, William would need to cross 6 roads ( $3 - 5 - 4 - 2 - 4 - 5 - 3$ ).

In the **second sample case** it could happen to have the contest in the 4-th city and the server parts in the 5-th city.

<sup>1</sup>In other words, each city is connected with at most 2 other cities.

## GitUbe Graph (gitube)

The Italian Olympic staff is very active on *GitUbe*, a website that is used by a lot of developers around the world to collaborate on open source code. William likes GitUbe so much that he wrote a “GitUbe-like” system, for his personal use. Sadly, a key feature is missing in William’s system: the *GitUbe Graph*.

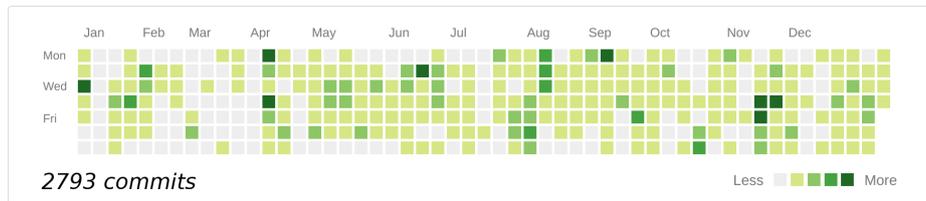


Figure 1: A sample *GitUbe Graph*.

The “GitUbe Graph” is a neat way to show users’ activity. For each user, the graph shows a grid where each cell is colored with some color depending on the number of *commits* made by the user in that day (e.g. dark green for a very active day, light green for a less active day, and so on). The grid has 7 rows (one for each day of the week) and some amount of columns (depending on how many days we want to show; GitUbe, for example, always shows the whole year).

For the purpose of this task, we can assume:

- We want the graph to show  $D$  days (numbered from 1 to  $D$ ).
- The first day (or “day number 1 of  $D$ ”) is always a Monday.
- The week starts on Monday.
- Like GitUbe, we will number the days in the graph going column by column, starting from the top left cell of the grid and ending in the bottom right cell.
- To better support colorblind people, we won’t use colors at all, and instead use special symbols:
  - The ‘.’ symbol, when there are less than 10 commits in the day.
  - The ‘o’ letter, when there are from 10 to 19 commits in the day.
  - The ‘0’ letter, when there are 20 or more commits in the day.

This is an example graph:

```

0o00oo.o000.00.00o..0..0.00.0.ooo0.ooo..
o00000000000.o..o0o0o0oo0.0..0o.o0.ooo.o0
.o..o0.o.0o.0o..o..000.0.0.0.o0.o.o.o.o00
..00o00.0.0o0o0o..00..o..o0ooo00.o00o0.
.0o0o0o0ooo00..ooo0ooo.o0...o.00.0o0.o0
000ooo00ooo0.o.0.o.o000oo...00..000o..0.
00.00ooo00...0.oo.o.00..o...0o..0.o.o00
    
```

You will be given the number of days  $D$ , the number of commits  $N$  made by some user, and then the actual list of commits. For each commit, you are given a single integer  $C_i$  which represents the day in which the commit was made.

Help William compute the user graph!

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

## Input

The first line contains two integers  $D$  and  $N$ . The second line contains  $N$  integers  $C_i$ .

## Output

You need to print the user graph as previously described.

## Constraints

- $1 \leq D \leq 366$  (it might be a leap year, or a 1-day long period, and so on).
- $1 \leq N \leq 100\,000$ .
- $1 \leq C_i \leq D$  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]:**  $N \leq 1000$ .
- **Subtask 3 [40 points]:**  $D \leq 7$ .
- **Subtask 4 [30 points]:** No additional limitations.

## Examples

input.txt	output.txt
<pre>20 37 4 8 15 5 5 13 13 5 3 11 13 13 16   18 5 13 13 1 13 18 13 13 5 5 12     4 5 8 5 6 4 14 9 13 5 1 5</pre>	<pre>... ... ... ... o.. .o. ..</pre>
<pre>7 40 1 2 6 2 2 4 2 2 1 2 6 2 6 1 4 6 2   7 2 2 2 2 7 3 2 3 1 2 2 4 2 2 2     4 2 2 6 4 5 2</pre>	<pre>. 0 . . . . .</pre>

## Winning Strategies (magnamagna)

Giorgio and William enjoy playing together the *Magnamagna* game, a typical Italian two-player game inspired by the well-known practice.

After collecting bribes from contestants<sup>2</sup>, they first arrange them into a long line of  $N$  items of nominal value  $V_i$  for  $i = 0 \dots N - 1$  (from left to right). Then, they play in turns each time choosing between two possibilities:

- take the leftmost item and gain its value ('L'),
- take the rightmost item and gain its value ('R').



Figure 1: The long line of bribes, to be taken in turns.

In any case, taking items from the interior of the line is never allowed.

Thanks to months of exhaustive research, Giorgio claims to have finally found the optimal strategy for this game, allowing him to collect the most possible money on every play. He thus decided to let William play first so as to compensate the strategic gap. Giorgio's strategy is indeed very simple: each time he chooses the move with the highest *smartness*. The smartness of a move is the quantity:

$$x_0 - \frac{x_1}{2} + \frac{x_2}{4} - \frac{x_3}{8} + \dots$$

where  $x_0, x_1, \dots$  are the values in the line of items, taken **from the chosen end** on.

For example, consider the following line of items:

6 3 0 8

In this case, the smartness of move 'L' is  $6 - 3/2 + 0/4 - 8/8 = 3.5$  and the smartness of move 'R' is  $8 - 0/2 + 3/4 - 6/8 = 8$ . Thus in this case, Giorgio would choose move 'R'. In the very unlikely case of the two moves having exactly the same smartness, Giorgio will let William choose for him.

Real numbers closer than  $10^{-6}$  are considered equal for the purpose of this task (we recommend to use double precision floating point numbers).

Despite Giorgio's claims, William is not that convinced of the optimality of this strategy. Help William collect the highest possible value, knowing that Giorgio always follows the above strategy!

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

### Input

The first line contains the only integer  $N$ . The second line contains  $N$  integers  $V_i$ .

<sup>2</sup>Any reference to people and facts is purely fictional.



## Output

You need to write a single line with a sequence of  $N$  characters ‘L’ or ‘R’, representing the moves taken by William and Giorgio.

## Constraints

- $1 \leq N \leq 1000$ .
- $0 \leq V_i \leq 100\,000$  for each  $i = 0 \dots N - 1$ .

## Scoring

Your program will be tested against several test cases grouped in subtasks. The score in each subtask will be calculated as the **minimum** score obtained in any of its test cases, multiplied by the value of the subtask. The score in a test case will be **0** if Giorgio’s moves in the produced sequence do not follow the strategy described above. Otherwise, it will be calculated as:

$$\max\left(\frac{W - W_{\text{avg}}}{W_{\text{opt}} - W_{\text{avg}}}, 0\right)$$

where  $W$  is the value collected by William through your program,  $W_{\text{opt}}$  is the maximum value that William can possibly collect, and  $W_{\text{avg}}$  is the value collected by William in average.<sup>3</sup>

- **Subtask 1 [ 5 points]**: Examples.
- **Subtask 2 [35 points]**:  $N \leq 10$ .
- **Subtask 3 [40 points]**:  $N \leq 100$ .
- **Subtask 4 [20 points]**: No additional limitations.

## Examples

input.txt	output.txt
4 6 3 0 8	RLLR
5 84 75 42 25 51	LLRLR

## Explanation

In the **first sample case**, William gains  $8 + 3 = 11$  and Giorgio gains  $6 + 0 = 6$ .

In the **second sample case**, William gains  $84 + 51 + 25 = 160$  and Giorgio gains  $75 + 42 = 117$ .

<sup>3</sup>The average is computed assuming that William chooses ‘L’ with probability 50% and ‘R’ also with probability 50%.





## 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]:  $N, G_i \leq 100$  for all  $i$ .
- **Subtask 3** [40 points]:  $G_i \leq 1000$  for all  $i$ .
- **Subtask 4** [20 points]: No additional limitations.

## Examples

input.txt	output.txt
6 1 10 10 5 10 5	2
11 7 4 3 7 2 5 7 5 2 6 5	4

## Explanation

In the **first sample case**, the trends are genres 1 and 10 which were both trending in time  $i = 2$ . Even though 5 is overall more frequent than 1, it has never been a trend.

In the **second sample case**, the trends are genres 3, 4, 7 (in time  $i = 3$ ) and 5 (in time  $i = 11$ ).

## Rescaling Sequence (rescaling)

Giorgio is working on a new research paper, this time about integer sequences. Today, he's looking for a specific kind of sequence: the *rescaling sequence*.

A rescaling sequence is a sequence of integers such that, for each pair of adjacent elements, one of the following statements is true:

- The second element is smaller than the first element.
- The second element is a multiple of the first element.

So, this is a rescaling sequence: 4, 8, 7, 21, 19. This however is *not*: 4, 8, 7, 20, 19 because 20 is not a multiple of 7.

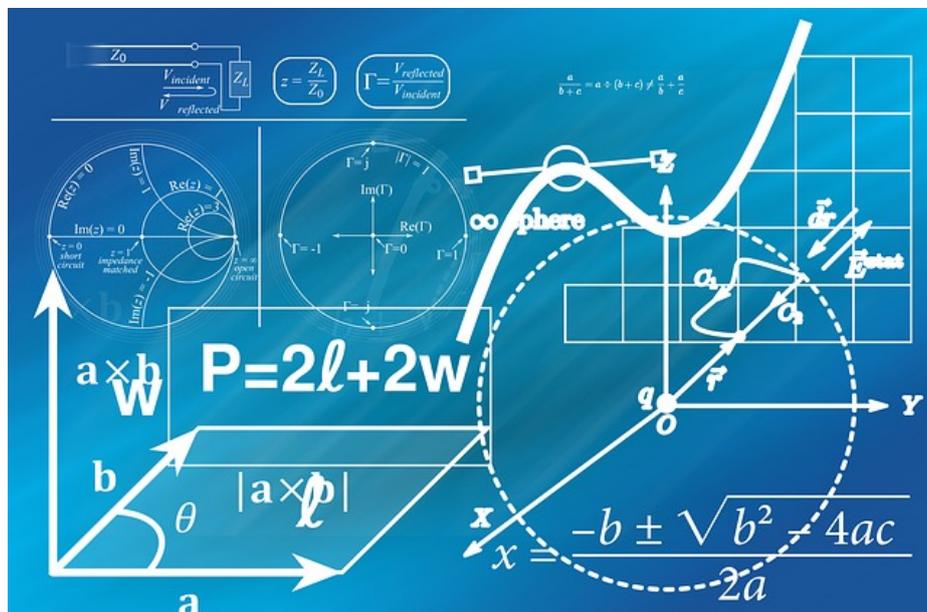


Figure 1: What Giorgio's whiteboard looks like.

You are given a sequence of  $N$  integers. Giorgio can delete some elements from the sequence, but he wants to delete as few as possible of them (possibly zero!). Help Giorgio obtain a rescaling sequence by computing the minimum number of elements to be deleted.

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

### Input

The first line contains the only integer  $N$ . The second line contains  $N$  integers  $S_i$ .

### Output

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



## Constraints

- $1 \leq N \leq 5000$ .
- $1 \leq S_i \leq 1\,000\,000$  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** [30 points]:  $N \leq 100$ .
- **Subtask 4** [20 points]: All the elements of the sequence are *prime numbers*.
- **Subtask 5** [20 points]: No additional limitations.

## Examples

input.txt	output.txt
5 4 8 7 21 19	0
5 4 8 7 20 19	2

## Explanation

The **first sample case** is the one described above.

In the **second sample case**, it is sufficient to erase either 8, 7 or 20, 19.