Dancing Elephants

Dancing Elephants is a spectacular show in Pattaya that features **N** elephants dancing on a line, known as the *stage*.

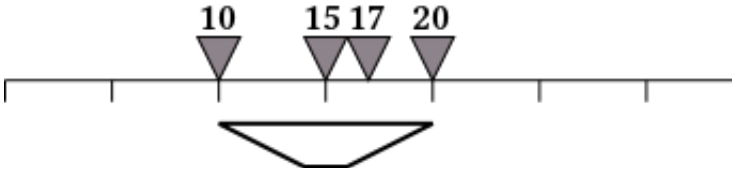
After years of training, elephants in this show are capable of many amazing dances. The show consists of a series of acts. In each act, exactly one elephant performs a cute dance while possibly moving to a different position.

The show producers want to produce a photo book that contains pictures of the entire show. After each act, they want to take pictures of all elephants as seen by the spectators.

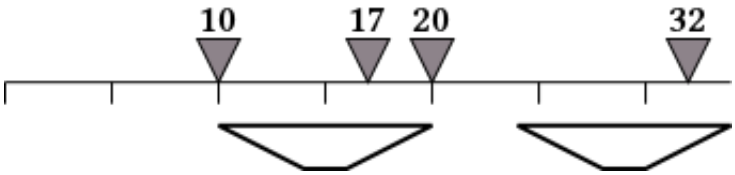
At any time during the show, multiple elephants may share the same position. In that case, they simply stand behind one another at the same position.

A single camera can take a picture of a group of elephants if and only if all their positions lie on some segment of length **L** (including both its endpoints). As the elephants can spread out across the stage, multiple cameras may be needed in order to take simultaneous snapshots of all the elephants.

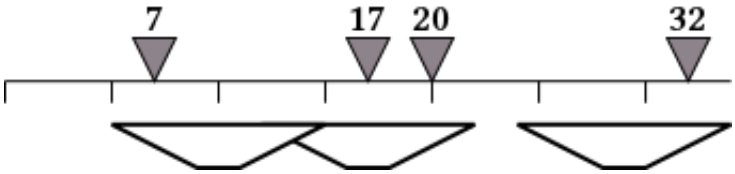
As an example, suppose that **L=10** and that the elephants are at positions 10, 15, 17, and 20 on the stage. At this moment, a single camera can take their picture, as shown below. (Elephants are shown as triangles; cameras are shown as trapezoids.)



In the following act, the elephant at position 15 dances to position 32. After this act, we need at least two cameras to take the snapshot.



In the next act, the elephant at position 10 moves to position 7. For the new arrangement of elephants, we need three cameras to photograph all of them.



In this interactive task, you have to determine the *minimum* number of cameras needed to take the pictures after each of the acts. Note that the number of cameras needed may increase, decrease, or stay the same between acts.

Your task

Write the following procedures:

- Procedure **init(N,L,X)** that takes the following parameters:
  - **N** – the number of elephants. The elephants are numbered **0** through **N-1**.
  - **L** – the length of the segment captured by a single camera. You may assume that **L** is an integer such that **0 ≤ L ≤ 1 000 000 000**.
  - **X** – a one-dimensional array of integers representing the initial positions of the elephants. For **0 ≤ i < N**, elephant **i** starts at the position **X[i]**. The initial positions are in sorted order. More precisely, you may assume that **0 ≤ X[0] ≤ ... ≤ X[N-1] ≤ 1 000 000 000**. Note that during the dance the elephants may reorder themselves.

  This procedure will be called only once, prior to all calls to **update**. It does not return any value.
- Procedure **update(i,y)** that takes the following parameters:
  - **i** – the number of the elephant that moves in the current act.
  - **y** – the position where the elephant **i** will stand after the current act. You may assume that **y** is an integer such that **0 ≤ y ≤ 1 000 000 000**.

  This procedure will be called multiple times. Each call corresponds to a single act (which follows on from all of the previous acts). Each call must return the *minimum number of cameras needed* to photograph all elephants after the corresponding act.

Example

Consider the case where **N=4**, **L=10**, and the initial positions of the elephants are

X=
10
15
17
20

First, your procedure **init** will be called with these parameters. Afterwards, your procedure **update** will be called once for each act. Here is an example sequence of calls and their correct return values:

act	call parameters	return value
1	update(2,16)	1
2	update(1,25)	2
3	update(3,35)	2
4	update(0,38)	2
5	update(2,0)	3

Subtasks

Subtask 1 (10 points)

- There are exactly **N** = 2 elephants.
- Initially, and after each act, the positions of all elephants will be distinct.
- Your procedure **update** will be called at most **100** times.

Subtask 2 (16 points)

- 1 ≤ **N** ≤ 100.
- Initially, and after each act, the positions of all elephants will be distinct.
- Your procedure **update** will be called at most **100** times.

Subtask 3 (24 points)

- 1 ≤ **N** ≤ 50 000.
- Initially, and after each act, the positions of all elephants will be distinct.
- Your procedure **update** will be called at most **50 000** times.

Subtask 4 (47 points)

- 1 ≤ **N** ≤ 70 000.
- Elephants may share the same position.
- Your procedure **update** will be called at most **70 000** times.

Subtask 5 (3 points)

- 1 ≤ **N** ≤ 150 000.
- Elephants may share the same position.
- Your procedure **update** will be called at most **150 000** times.
- Please see the note about the CPU time limit under the Implementation Details Section.

Implementation details

Limits

- CPU time limit: 9 seconds
  Note: The collection templates in the C++ Standard Library (STL) can be slow; in particular, it might not be possible to solve subtask 5 if you use them.
- Memory limit: 256 MB
  Note: There is no explicit limit for the size of stack memory. Stack memory counts towards the total memory usage.

Interface (API)

- Implementation folder: elephants/
- To be implemented by contestant: elephants.c or elephants.cpp or elephants.pas
- Contestant interface: elephants.h or elephants.pas
- Sample grader: grader.c or grader.cpp or grader.pas
- Sample grader input: grader.in.1, grader.in.2, ...

  Note: The sample grader reads the input in the following format:

  - Line 1: **N**, **L**, and **M**, where **M** is the number of acts in the show.
  - Lines 2 to **N+1**: the initial positions; i.e., line **k+2** contains **X[k]** for $0 \le k < N$.
  - Lines **N+2** to **N+M+1**: information on **M** acts; i.e. line **N+1+j** contains **i[j]**, **y[j]**, and **s[j]**, separated by a space, denoting that in the **j**-th act elephant **i[j]** moves to position **y[j]**, and after that act, **s[j]** is the mininal number of cameras needed, for $1 \le j \le M$.

- Expected output for sample grader input: grader.expect.1, grader.expect.2, ...
  For this task, each one of these files should contain precisely the text "**Correct.**"