A Light Inconvenience (light)

The Scientific Committee is relaxing at the opening ceremony of this year's CEOI. The tasks are prepared, the grading server's 10^{12} firewalls are finally up, and the committee is looking forward to an amazing show with flaming torches. Nothing can go wrong. Except... no one bought enough oil for those torches! Now the committee needs help to run the show without using up their oil supplies.

During the show, there will be performers standing in a line, numbered from left to right starting at 1. The number of performers will vary over time. Each performer holds a torch which may or may not be on fire at any point in time. Initially, there is only one performer whose torch is on fire.

The show is divided into Q acts. At the beginning of act a, either $p_a > 0$ performers decide to join the line on the right, or the rightmost $p_a > 0$ performers decide to leave. This is out of the committee's control. The leftmost performer always remains on stage. The torches of joining performers are not on fire, and leaving performers extinguish their torches if they are on fire.

As soon as the line of performers for act a is ready, the committee has to announce a number $t_a \geq 0$. Then, each performer whose torch is on fire shares their fire with the t_a performers on their right. This means that the torch of performer i will be on fire afterwards if and only if the torch of at least one of the performers $\max\{i - t_a, 1\}, \dots, i$ was on fire beforehand. *For a more dynamic show, t_a must not be larger than $5p_a$, and should be as low as possible* (see the grading section below).

At the end of each act, the committee has to tell every performer whose torch is currently on fire whether to extinguish it or not. For aesthetic reasons, the torch of the rightmost performer should always be on fire after that. *Moreover, to conserve oil, the number of torches left on fire must not be larger than* 150.

Write a program that tells the committee how to run the show within these constraints.

Communication

This is a communication task. You must implement the following three functions:

- **void** *prepare*() is called in the beginning. You can use this function for setup (or do nothing).
- **pair⟨long long, vector⟨long long⟩⟩** *join*(**long long** p_a) is called when $p_a > 0$ performers join the line on the right. This function must return a pair that consists of the number t_a announced by the committee and the list of those performers whose torches should be on fire at the end of the act. *This list must be in strictly increasing order.*
- **pair⟨long long, vector⟨long long⟩⟩** *leave*(**long long** p_a) is called when the rightmost $p_a > 0$ performers leave the line. Again, it must return a pair that consists of the number t_a announced by the committee and the list of those performers whose torches should be on fire at the end of the act, in strictly increasing order.

If any of your return values does not satisfy the above constraints, your program will be immediately terminated and judged as **Not correct** for the respective testcase. You must not write to standard output or read from standard input; otherwise, you may receive the verdict **Security violation!**. You are however free to write to the standard error stream (*stderr*).

You must include the file `light.h` in your source code. To test your program locally, you can link it with `sample_grader.cpp`, which can be found in the attachment for this task in CMS. See below for a description of the sample grader, and see `sample_grader.cpp` for instructions on how to run it with your program. The attachment also contains a sample implementation as `light_sample.cpp`.

Constraints and grading

Let N denote the maximum number of performers who are simultaneously standing in the line at any time. In all testcases, $N \leq 10^{17}$ and $1 \leq Q \leq 50\,000$.

Subtask 1 (5 points).   There is only one call to *leave* per testcase.

Subtask 2 (5 points). $N \leq 700$

Subtask 3 (10 points). $N \leq 5\,000$

Subtask 4 (5 points). $N \leq 25\,000$

Subtask 5 (10 points). $N \leq 100\,000$

Subtask 6 (5 points). $N \leq 500\,000$

Subtask 7 (60 points). No further constraints.

Partial scoring. In Subtask 7, your actual score depends on the maximum value of t_a/p_a over all acts a according to the following table:

$\max t_a/p_a$	$[0,1]$	$(1,2]$	$(2,3]$	$(3,5]$
score	60	35	20	10

In particular, to get full score all return values of *join* and *leave* must satisfy $t_a \leq p_a$.

Example interaction

Consider a testcase with $Q = 4$. Here, an interaction between your program and the grader could look as follows:

Call	Return value	Explanation
prepare()	—	you're free to do any setup here (or do nothing)
		the ceremony starts with one performer whose torch is on fire
		🔥
join(3)	3, {2, 4}	three performers join for a total of four performers
		performer 1 lights the torches of performers 2, 3, and 4; afterwards, performers 1 and 3 extinguish their torches
		┆ 🔥 ┆ 🔥
leave(2)	0, {2}	the two rightmost performers leave (only two remain)
		no new torches are lit; the torch of performer 2 remains on fire
		┆ 🔥
join(2)	3, {2, 4}	two new performers join (for a total of four)
		performer 2 lights the torches of performers 3 and 4; performer 3 extinguishes their torch afterwards
		┆ 🔥 ┆ 🔥
join(3)	3, {2, 4, 7}	three new performers join (for a total of seven)
		first the torches of performers 3, 5, 6, and 7 are lit, then performers 3, 5, and 6 extinguish their torches again
		┆ 🔥 ┆ 🔥 ┆ ┆ 🔥
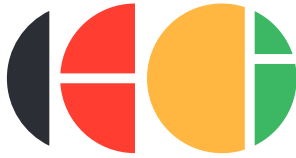
The above interaction is reproduced by `light_sample.cpp` on the public testcase.

Note that the above sequence of calls to *join* and *leave* would constitute a valid testcase in any of the subtasks.

Grader

The sample grader first expects on standard input the integer Q. It then writes to standard output a protocol of all calls to the above three functions.

In the beginning, the grader calls *prepare*( ). Then it simulates the Q acts. For each act a, it expects on standard input either an integer $p_a > 0$, meaning that p_a performers join the line, or an integer $q_a < 0$, meaning that $p_a := -q_a$ performers leave the line; it will then call *join*(p_a) or *leave*(p_a), respectively.

Upon termination, it writes one of the following messages to standard output:

Invalid input. The input to the grader via standard input was not of the above form.

The stage is empty. There is less than one performer left.

Invalid return value. The return value t_a does not satisfy $0 \le t_a \le 5p_a$, the list of performers was not given in strictly increasing order, or the list of performers contains an integer smaller than 1 or larger than the total number of performers on stage.

Too many burning torches. More than 150 torches were on fire after the act.

Rightmost torch not on fire. The rightmost torch was not lit, or it was extinguished.

Not all announced torches have been lit. The torch of at least one of the performers returned by your function has not been lit.

Correct: ratio at most f, at most b burning torches. None of the above errors occurred, all calls to *leave* and *join* satisfied $t_a \le f \cdot p_a$ (up to rounding errors), and there were at most b torches on fire after any act.

In contrast, the grader actually used to judge your program will only output **Not correct** (for any of the above errors), **Security violation!**, **Partially correct**, or **Correct**. Moreover, the grader is *adaptive*, i.e. the number of performers joining or leaving in each act may depend on the behavior of your program in the current as well as in earlier runs. Both the sample grader and the grader used to judge your program will terminate your program automatically whenever one of the above errors occurs.

Limits

Time: 1 s
Memory: 512 MiB