Problem C. Convex Hull

- 2023.10.06 22:40 Update: Added sample grader interaction.

Problem Description

An upper convex hull denoted as $\mathcal{C}$ comprises $L+1$ lattice points, specifically $\{(0, y[0]), (1, y[1]), \ldots, (L, y[L])\}$.

However, $\mathcal{C}$ is not directly visible. Therefore, in order to observe $\mathcal{C}$, you need to submit queries of the following type:

- Request a slope m with an integer value, and we will provide you with the tangent point where the line $y = mx + c$ intersects $\mathcal{C}$ from above (in cases where there are multiple such points, we will furnish you with the one having the smallest x coordinate).

It is essential to note that $y[0] = y[L] = 0$, and for $i = 1, 2, \ldots, L-1$, the values of $y[i]$ are bounded within the range $1 \leq y[i] \leq 10^9$. Your objective is to determine the value of $y[k]$.

Please be aware that the convex hull may **NOT** exhibit strict convexity.

What is an upper convex hull?

An "upper convex hull" refers to the boundary that encloses the uppermost points of a set of data in a way that creates a convex shape. In a two-dimensional space, it is the smallest convex polygon that encompasses the highest points of a given dataset.

Implementation Details

You should implement the following procedure:

```
int convex_hull(int L, int k)
```

- L: the upper convex hull $\mathcal{C}$ comprises $L+1$ lattice points.
- k: your objective is to determine the value of $y[k]$.
- This procedure should return the value of $y[k]$.
- This procedure will be called at most t times.

You can call the following procedure at most 32 times per call to `convex_hull`:

```
pair<int, int> query(int m)
```

- m: the slope you request, the value must be within $[-10^9, 10^9]$.

- This procedure will return the coordinates of the leftmost point where the line $y = mx + c$ tangentially touches the upper convex hull $\mathcal{C}$ from above.

Constraints

- $1 \leq t \leq 20$.
- $2 \leq L \leq 10\,000$.
- $1 \leq k \leq L - 1$.
- $1 \leq y[i] \leq 10^9$ for $i = 1, 2, \ldots, L - 1$.
- $y[0] = y[L] = 0$.
- Points $\{(0, y[0]), (1, y[1]), \ldots, (L, y[L])\}$ form an upper convex hull.

Subtasks

1. (40 points) Any three points do not collinear.
2. (60 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1	1000	262144
1	2-5	1000	262144
2	1-11	1000	262144

Examples

Example 1

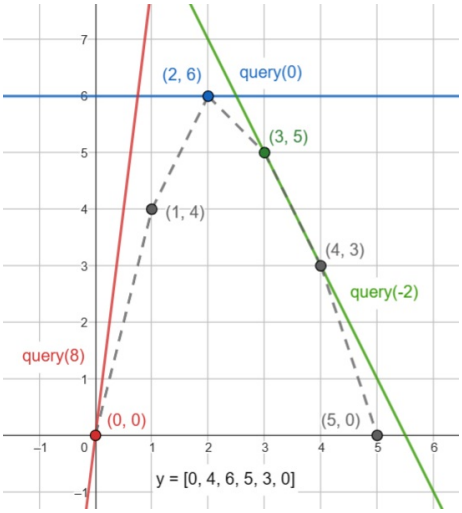
Consider a scenario with $y = [0, 4, 6, 5, 3, 0]$ and $k = 3$:

```
convex_hull(5, 3)
```

The interaction **MAY** proceeds as follows:

1. We start a query by calling `query(0)`. This query returns the point $(2, 6)$, which is shown in blue below.
2. Next, we make a query with `query(8)`, resulting in the point $(0, 0)$, illustrated by the red line.
3. Lastly, we call `query(-2)`, which gives us the point $(3, 5)$, represented by the green line.

Based on these queries, we deduce that $y[3] = 5$. Therefore, the `convex_hull` procedure returns the value 5 as the result.



The upper convex hull $\mathcal{C}_1$.

This sample input satisfies the constraints of all the subtasks.

Example 2

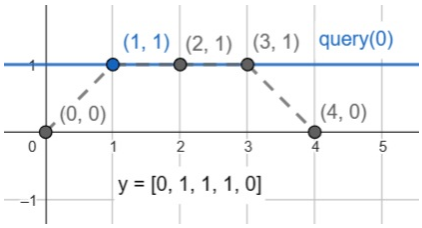
Let's consider another scenario with $y = [0, 1, 1, 1, 0]$ and $k = 2$:

```
convex_hull(4, 2)
```

Here's how the interaction **MAY** unfolds:

1. We initiate a query with `query(0)`. This query yields the point $(1, 1)$, as seen in blue below.

By observing that the highest y-value among these points is 1, and since $y[2] \neq 0$, we conclude that $y[2] = 1$. Consequently, the procedure returns the value 1 as the result.



The upper convex hull $\mathcal{C}_2$.

This sample input satisfies the constraints of Subtask 2.

Example 3

Let's consider another scenario with $y = [0, 9, 17, 21, 24, 26, 27, 0]$ and $k = 4$:

```
convex_hull(7, 4)
```

The procedure should return 24.

This sample input satisfies the constraints of all the subtasks.

Sample grader

The sample grader reads the input in the following format:

- line 1: t
- line $2+2i$ ($0 \le i \le t-1$): L k
- line $3+2i$ ($0 \le i \le t-1$): $y[0]$ $y[1]$ $\ldots$ $y[L]$

The sample grader prints your queries in the following format:

- `<k>th query: slope = <m>`: where `<k>` denotes the k^{th} call to `query`, and `<m>` denotes the parameter m you passed into `query`.

You should input the x coordinate which the function should return.

For each call to `convex_hull`, the sample grader prints `convex_hull: <y>` in the first line, where `<y>` is the return value of `convex_hull`.

If the answer is correct, the sample grader prints `Accepted: <query_count>` in the second line, where `<query_count>` is the number of calls to `query` you make.

If the answer is incorrect, the sample grader prints `Wrong Answer: <MSG>` in the second line, where `<MSG>` is one of the following:

- `wrong coordinate`: the return value of `convex_hull` is not $y[k]$.

The following describes the scenario in Example 1.

User Input	Grader Output	Notes
`1`		$t = 1$
`5 3`		First testcase: $(L, k) = (5, 3)$
`0 4 6 5 3 0`		$y = [0, 4, 6, 5, 3, 0]$
	`1st query: slope = 0`	Your program calls `query(0)`.
`2`		It should return $(2, y[2])$.
	`2nd query: slope = 8`	Your program calls `query(8)`.
`0`		It should return $(0, y[0])$.
	`3rd query: slope = -2`	Your program calls `query(-2)`.
`3`		It should return $(3, y[3])$.
	`convex_hull: 5`	Your program returns $y[k] = 5$.
	`Accepted: 3`	The answer is correct, and you have used 3 queries.

If the sample grader detects a protocol violation, the output of the sample grader is `Protocol Violation: <MSG>`, where `<MSG>` is one of the following:

- `too many queries`: `query` is called more than 32 times in any call to `convex_hull`.
- `invalid parameters`: `query` is called with m not within $[-10^9, 10^9]$.

Please note that the sample grader does **NOT** validate the input (e.g. check the convexity of $\mathcal{C}$).

Notes

- Here is a sample implementation. (Link)
- You should include `"1612.h"` in your program.
- You should **NOT** implement the `main` function.
- You should only submit `1612.cpp` to the Online Judge.
- You should **NOT** read anything from `stdin` or print anything to `stdout`.
- You can use `stderr` for debug (`std::cerr`).
- You can modify the grader as you want.
- You can use `g++ -std=c++17 -O2 -o 1612 1612.cpp grader.cpp` to compile the code, and use `./1612` or `1612.exe` to run the code.