3 Good Bunny Score

3.1 Problem Statement

Supreme Leader Armpit has imprisoned N citizens in his torture chamber. They are arranged in a line and numbered from 1 to N. Citizen i has a *Good Bunny Score*, S_i, denoting how good of a bunny they are. It is guaranteed that **no** two citizens will have the same score.

Supreme Leader Armpit will torture Q citizens. Each time he wants to torture a citizen, he will choose two numbers, l and r. For the citizens numbered with some $x \in [l, r]$, he will torture the one with the lowest Good Bunny Score.

Unfortunately for Supreme Leader Armpit, the previous mayor, Benson the Wabbit, designed the Good Bunny Score system to be hard to access. The system storing the Good Bunny Scores will only answer one type of query: given i and j, determine if citizen i or citizen j has a lower Good Bunny Score.

Supreme Leader Armpit will be very displeased if he tortures the wrong citizen. Thus, the task falls on you once again to determine who should be tortured.

3.2 Implementation Details

Your code should contain `#include "bunnyscore.h"`. It should not read from standard input or write to standard output as doing so will give `Checker error` verdict (ask a clarification if you do not know why you get the verdict). You should implement the follow procedures:

`void init(int N,int Q,int K,int T)`

- N: the size of the permutation
- K: the maximum number of calls to procedure `smaller`
- Q: the number of calls to procedure `query`
- T: the subtask number

`int query(int l, int r)`

- This function should return the index with the minimum value in the range $[l, r]$ (both inclusive)
- $1 \leq l \leq r \leq N$
- This function will be called exactly Q times

You may make calls to the following procedure (you are allowed to call this during `query`):

`bool smaller(int i, int j)`

- $1 \leq i, j \leq N$
- This function returns `true` if $S_i < S_j$ and `false` otherwise.

3.3 Subtasks

For all testcases, it is guaranteed that:

- $1 \leq N \leq 100000$
- $1 \leq Q \leq 1000000$
- $S_i \neq S_j$ for all $i \neq j$

Subtask	Score	N, Q	K	T
0	5	$N, Q \leq 1000$	$K = 10^6$	$T = 1$
1	95	-	$K = 10^8$	$T = 2$

Note: the grader is **not** adaptive. That is the array S is fixed before the procedure `init` is called.

3.4 Scoring

If in any of the test cases, the calls to the procedure `smaller` do not conform to the rules mentioned above, or the return value of `query` is incorrect, the score of your solution will be 0.

For subtask 2, you can get a partial score. let M be the number of calls to the procedure `smaller` in a single testcase. Then, the score for that testcase will be calculated according to the following table. Note that your score will be the minimum score over all testcases.

Condition	Score
$10^8 < M$	0
$5 \cdot 10^6 < M \leq 10^8$	15
$200000 \leq M \leq 5 \cdot 10^6$	$200 \cdot (1 + e^{\frac{M}{2000000}})^{-1}$
$M < 200000$	95

For reference, there is a table of points for subtask 2 for various values of M:

M	Points
200000	95.00
400000	90.03
600000	85.11
800000	80.26
1000000	75.51
2000000	53.79
3000000	36.49
4000000	23.84
5000000	15.17

3.5 Sample Interaction

Let $N = 5$, $Q = 2$, $K = 10$, $T = 1$ and $S = \{3, 1, 4, 15, 9\}$.

The procedure `init` is called below:

```
init(5,2,10,1)
```

This procedure may call `smaller(2,4)`, which will return `true`. It may also call `smaller(5,1)` which will return `false`.

Then the first call to `query` is called:

```
query(3,3)
```

The element with the smallest element is clearly 3, so the procedure `query(3,3)` returns 3. Then the second call to `query` is called:

```
query(1,2)
```

The procedure `query` can call `smaller(1,2)` which returns `false`. The procedure `query` then concludes that the smallest element is 2 and returns 2.

3.6 Sample Grader

A sample grader is provided as attachment. The sample grader reads input in the following format:

- line 1: $N\ Q\ K\ T$
- line 2: $S_1\ S_2\ \ldots\ S_N$
- line $3+i$ $(0 \leq i < Q)$: $l_i\ r_i$

If your program is judged as `Accepted`, the sample grader prints output in the following format:

- line $1+i$ $(0 \leq i < Q)$: ans_i, where ans_i is the answer to the i-th query.
- line $Q+1$: `Queries used:` q, where q is the total number of queries used.

If you program is judged as `Wrong Answer`, the sample grader prints `Wrong Answer: MSG`. The meaning of `MSG` is as follows:

- `Invalid query`: In procedure `smaller`, the condition $1 \leq i, j \leq N$ is not satisfied.
- `Too many queries`: The procedure `smaller` is called more than K times.