Problem I. Classical Data Structure Problem

Problem Description

You are given a set $S = \{a[0], a[1], \ldots, a[n-1]\}$, and you will have to support q operations of three types on this set:

1. Remove the smallest integer from S. It is guaranteed that $S \neq \emptyset$.
2. Remove the largest integer from S. It is guaranteed that $S \neq \emptyset$.
3. Insert $\mathsf{MEX}(S)$ into S.

Define $\mathsf{MEX}(S)$ as the smallest non-negative integer that has not appeared in S.

Implementation Details

You should implement the following procedures:

```
void init(int n, int[] a)
```

- n: the initial size of the set S.
- a: arrays of length n. For $0 \leq i \leq n-1$:
  - $a[i]$ is the i^{th} element in S.
- This procedure is called exactly once, before any calls to `remove_min`, `remove_max`, and `insert_mex` (see below).

```
int remove_min()
```

- This procedure should return the minimum integer in S, $\mathsf{MIN}(S)$.
- It is guaranteed that $S \neq \emptyset$ when calling this function.
- After calling this function, $\mathsf{MIN}(S)$ should be removed from set S.

```
int remove_max()
```

- This procedure should return the maximum integer in S, $\mathsf{MAX}(S)$.
- It is guaranteed that $S \neq \emptyset$ when calling this function.
- After calling this function, $\mathsf{MAX}(S)$ should be removed from set S.

```
int insert_mex()
```

- This procedure should return the minimum non-negative integer not in S, $\mathsf{MEX}(S)$.
- After calling this function, you should insert $\mathsf{MEX}(S)$ into set S.

- The total number of calls to `remove_min`, `remove_max`, and `insert_mex` is exactly q.

Constraints

- $0 \le n \le 200\,000$.
- $1 \le q \le 10\,000\,000$.
- $0 \le a[i] \le 10^9$ for $i = 0, 1, \ldots, n-1$.
- $a[i] \ne a[j]$ for $i \ne j$.

Subtasks

1. (1 point) $q \le 5000$.
2. (9 points) $q \le 1\,000\,000$.
3. (20 points) $q \le 3\,000\,000$.
4. (70 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
1	1-7	1000	262144
2	1-14	1000	262144
3	1-21	500	262144
4	1-28	500	65536

Examples

Consider the following call:

```
init(5, [4, 8, 7, 6, 3])
```

The initial set $S = \{4, 8, 7, 6, 3\}$.

Let's say $q = 7$, and the grader calls the following functions:

Function Call	S	Return Value
`remove_min()`	$\{4, 8, 7, 6\}$	3
`insert_mex()`	$\{4, 8, 7, 6, 0\}$	0
`remove_max()`	$\{4, 7, 6, 0\}$	8
`insert_mex()`	$\{4, 7, 6, 0, 1\}$	1
`remove_min()`	$\{4, 7, 6, 1\}$	0
`insert_mex()`	$\{4, 7, 6, 1, 0\}$	0
`remove_max()`	$\{4, 6, 1, 0\}$	7

As such, the procedure should return 3, 0, 8, 1, 0, 0, 7.

Sample grader

The sample grader reads the input in the following format:

- line 1: n q
- line 2: $a[0]$ $a[1]$ $\cdots$ $a[n-1]$
- line 3: $op[0]$ $op[1]$ $\cdots$ $op[q-1]$

Where $op[i] \in \{1, 2, 3\}$ ($0 \le i \le q-1$) denotes a call to `remove_min`, `remove_max`, or `insert_mex` respectively.

The sample grader prints your answers in the following format:

- line $1+i$ ($0 \le i \le q-1$): the i^{th} called function name, and the return value of the call.

Notes

- Here is a sample implementation. (Link)
- You should include `"1608.h"` in your program.
- You should **NOT** implement the `main` function.
- You should only submit `1608.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 use `g++ -std=c++17 -O2 -o 1608 1608.cpp grader.cpp` to compile the code, and use `./1608` or `1608.exe` to run the code.

Conventions

The task statements specify signatures using generic type names `void`, `string`, `int`, `int[]` (array), and `bool[][]` (2D array).

In each of the supported programming languages, the graders use appropriate data types or implementations, as listed below

Language	**void**	**string**	**int**	**int[]**	length of array **a**
C++	`void`	`std::string`	`int`	`std::vector<int>`	`a.size()`

A 2D array is a non-empty array of arrays of the same length.

Language	**bool[][]**	#rows in 2D array **a**	#columns in 2D array **a**
C++	`std::vector<std::vector<bool>>`	`a.size()`	`a[0].size()`