Problem Floppy

C header:	floppy_c.h
C++ header:	floppy.h

Roxette the old tech geek, stumbled upon an array $v_0, v_1, v_2, \ldots, v_{N-1}$ of N **distinct** integers. Finding it of great interest, Roxette wanted to save the array on her floppy disk. However, due to low free disk space, Roxette has to settle for less: she won't be able to save the whole array, and instead she plans on saving an array of bits that would allow her to answer any query of the following form:

$query(a, b) = id$, where $a \leq id \leq b$ and $v_{id} = max(v_a, v_{a+1}, \ldots, v_{b-1}, v_b)$

In other words, the query returns the index of the maximum value in a given sub-array.

Roxette is now asking for your help. Twice! First, she will provide you with the interesting array and you will have to tell her what sequence of bits to save on her floppy disk. Second, if she needs to know the answer to some queries, she will provide you with the sequence of bits you told her to save on the floppy disk and the queries she needs answered while you have to provide her with the correct answer to each of the queries.

Interaction

This is a two interactions task!
The contestant must implement two functions. The first of them is the following:

```
(C)   void read_array(int subtask_id, int N, int* v);
(C++) void read_array(int subtask_id, const std::vector<int> &v);
```

This function will be called **exactly once**, in the first interaction, and supplies the contestant with the array of great interest to Roxette. In the implementing of this function, the contestant must call the following function **exactly once**, which will be implemented by the grading program:

```
(C)   void save_to_floppy(int L, char* bits);
(C++) void save_to_floppy(const std::string &bits);
```

This function tells Roxette what sequence of bits to save on her floppy disk. Parameter `L` indicates the number of bits to be saved. Parameter `bits` must be a sequence of characters (only characters `'0'` and `'1'` are allowed).

The second function that the contestant must implement is:

```
(C)   int* solve_queries(int subtask_id,
                         int N, int L, char* bits,
                         int M, int* a, int* b);
(C++) std::vector<int> solve_queries(int subtask_id,
                                     int N, const std::string &bits,
                                     const std::vector<int> &a,
                                     const std::vector<int> &b);
```

This function will be called **exactly once**, in the second interaction, and supplies the contestant with the length of the array of great interest for Roxette, the array of bits Roxette was previously instructed by the contestant to save on her floppy disk and a list of M queries.

The parameters of the `i`th query are `a[i]` and `b[i]`.

This function must return an array of M integers, representing the answer to each of the queries (the `i`th integer must be the answer to the `i`th query).

Important: The contestant's program will run twice, once for each interaction. Therefore, any data computed by the contestant's program in the first run will not be accessible in the second run.

Constraints

- $-10^9 \leq v_i \leq 10^9$ for all $0 \leq i \leq N-1$
- $1 \leq L \leq 200\ 000$

Subtask 1 (7 points)

- $1 \leq N \leq 500$
- $0 \leq v_i < N$ for all $0 \leq i \leq N-1$
- $1 \leq M \leq 1\ 000$
- The score for the subtask will be 7 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 2 (21 points)

- $1 \leq N \leq 10\ 000$
- $1 \leq M \leq 20\ 000$
- The score of each test is $\min(1, \frac{1}{2^{\frac{L}{N}-1-\log_2 N}})$. The score for the subtask will be the minimum of the scores for each test, multiplied by 21.

Subtask 3 (72 points)

- $1 \leq N \leq 40\ 000$
- $1 \leq M \leq 80\ 000$
- The score of each test is $\min(1, \frac{1}{2^{\frac{L}{N}-2}})$. The score for the subtask will be the minimum of the scores for each test, multiplied by 72.

Interaction example

In the first interaction, the contestant's function will be called:

```
read_array(
  /* subtask_id = */ 3,
  /* v          = */ {40, 20, 30, 10});
```

This function, in turn, might choose to save the following bits to the floppy disk:

```
save_to_floppy(
  /* bits = */ "001100");
```

The first instance of the contestant's program will now be terminated, hence any data being stored in memory by this program will be lost.

In the second interaction, the contestant's function will be called:

```
solve_queries(
  /* subtask_id = */ 3,
  /* N          = */ 4,
  /* bits       = */ "001100",
  /* a          = */ {0, 0, 0, 0, 1, 1, 1, 2, 2, 3},
  /* b          = */ {0, 1, 2, 3, 1, 2, 3, 2, 3, 3});
```

This function should return an array of M integers:

```
  {0, 0, 0, 0, 1, 2, 2, 2, 2, 3}
```