Problem Floppy

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

Роксет — старомодная девушка-гик, недавно нашла массив $v_0, v_1, v_2, \ldots, v_{N-1}$, содержащий N **различных** целых чисел. Заинтересовавшись массивом, Роксет решила сохранить его на свою дискету. Однако, учитывая, что на дискете довольно мало места, она решила, вместо того, чтобы сохранять на дискету свой массив, сохранить туда массив битов, который бы позволил ей отвечать на запросы следующего вида:

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

Другими словами, запрос должен возвращать индекс максимального элемента на отрезке массива.

Роксет нужна ваша помощь. Причем дважды! Во-первых, она собирается выдать вам свой массив, а вы должны сказать ей, какую последовательность битов сохранить на дискету. Во вторых, она просит вас отвечать на запросы: она передаст вам последовательность битов, которую вы попросили ее сохранить на дискете, и запросы, на которые ей необходимо отвечать, а вы должны сообщить ей ответы на запросы.

Взаимодействие

Это задача, в которой решение будет запускаться дважды!
Участник должен реализовать две функции. Первая из них следующая:

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

Эта функция будет вызвана **ровно один раз**, во время первого запуска программы, она принимает в качестве аргумента массив, который нашла Роксет. При реализации этой функции необходимо **ровно один раз** вызвать следующую функцию в грейдере:

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

Эта функция сообщает Роксет, какую последовательность битов необходимо сохранить на дискету. Аргумент `L` задает число битов, которое нужно сохранить. Аргумент `bits` должен содержать эту последовательность, он должен состоять только из символов `'0'` и `'1'`.

Вторая функция, которую следует реализовать, следующая:

```
(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);
```

Эта функция будет вызвана **ровно один раз** во время второго запуска программы, она передает участнику длину массива, который был у Роксет, массив битов, который программа участника попросила Роксет записать на дискету, и массив из M запросов.

Параметры `i`-го запроса — `a[i]` и `b[i]`.

Функция должна вернуть массив из M целых чисел, задающих ответы на запросы (`i`-е число должно быть ответом на `i`-й запрос).

Важно: Программа участника будет запущена дважды. Оба запуска будут независимыми, не следует пытаться передавать данные любым образом между запусками.

Ограничения

- $-10^9 \leq v_i \leq 10^9$ для всех $0 \leq i \leq N-1$
- $1 \leq L \leq 200\,000$

Подзадача 1 (7 баллов)

- $1 \leq N \leq 500$
- $0 \leq v_i < N$ для всех $0 \leq i \leq N-1$
- $1 \leq M \leq 1\,000$
- Баллы за подзадачу равны 7, если все тесты пройдены, иначе 0.

Подзадача 2 (21 балл)

- $1 \leq N \leq 10\,000$
- $1 \leq M \leq 20\,000$
- Баллы за тест равны $\min(1, \frac{1}{2^{\frac{L}{N}-1-\log_2 N}})$. Баллы за подзадачу равны минимальному баллу за тест, умноженному на 21.

Подзадача 3 (72 балла)

- $1 \leq N \leq 40\,000$
- $1 \leq M \leq 80\,000$
- Баллы за тест равны $\min(1, \frac{1}{2^{\frac{L}{N}-2}})$. Баллы за подзадачу равны минимальному баллу за тест, умноженному на 72.

Пример запуска

При первом запуске будет вызвана функция

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

Эта функция может, например, сохранить следующий массив битов на дискету:

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

После этого первый запуск будет закончен, программа участника будет завершена, память будет очищена.

Во время второго запуска будет вызвана функция

```
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});
```

Эта функция должна вернуть массив из M целых чисел:

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