Problem Floppy

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

Roxette a dat peste un tablou $v_0, v_1, v_2, \ldots, v_{N-1}$ cu N **intregi distincti**. Găsindu-l foarte interesant, Roxette a vrut să salveze şirul pe dischetă. Cu toate acestea, din cauza spațiului liber scăzut pe dischetă, Roxette trebuie să se mulțumească cu mai puțin: nu va putea salva întregul şir și, în schimb, intenționează să salveze un şir de biți care să-i permită să răspundă la orice întrebare de forma următoare:

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

Cu alte cuvinte, răspunsul la o interogare este indicele valorii maxime dintr-o subsecvenţă.

Roxette va cere acum ajutorul. De două ori! În primul rând, ea vă oferă şirul interesant şi trebuie să îi spuneţi secvenţa de biţi care trebuie salvată pe dischetă. În al doilea rând, dacă are nevoie să știe răspunsul la unele interogări, vă va oferi secvența de biți pe care i-ați spus să o salveze pe dischetă și întrebările la care are nevoie de răspuns, în timp ce trebuie să îi oferiți răspunsul corect la fiecare dintre ele.

Interacţiune

Este o problemă cu două interacţiuni!
Concurentul trebuie să implementeze două funcţii. Prima este următoarea:

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

Prima va fi apelată **exact o dată**, în prima interacţiune, şi îi oferă concurentului şirul de mare interes pentru Roxette. În implementarea acestei funcții, concurentul trebuie să apeleze următoarea funcție **exact o dată**, care va fi implementată prin programul grader:

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

Această funcţie îi spune lui Roxette ce secvenţă de biţi să salveze pe dischetă.Parametrul `L` indică numărul de biţi care trebuie salvaţi. Parametrul `bits` trebuie să fie o secvenţă de caractere (numai caracterele `'0'` şi `'1'` sunt permise).

A doua funcţie pe care concurentul trebuie să o implementeze este:

```
(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);
```

Această funţie va fi apelată **exact o dată**, în a doua interacţiune, și furnizează concurentului lungimea şirului de mare interes pentru Roxette, şirul de biți care i-a fost dat antetior Roxette-ei să îl salveze pe discheta ei și o listă cu M interogări.

Parametrii celei de-a `i`$^{\text{a}}$ interogări sunt `a[i]` și `b[i]`.

Această funcție trebuie să returneze un șir de M întregi, reprezentând răspunsul la fiecare întrebare (al `i`$^{\text{-lea}}$ întreg va fi răspunsul la cea de-a `i`$^{\text{-a}}$ întrebare).

Important: Programul concurentului va rula de două ori, o dată pentru fiecare interacțiune. Prin urmare, orice date calculate de programul concurentului la prima rulare nu vor fi accesibile la a doua.

Restricții

- $-10^9 \leq v_i \leq 10^9$ pentru fiecare $0 \leq i \leq N-1$
- $1 \leq L \leq 200\ 000$

Subtask 1 (7 puncte)

- $1 \leq N \leq 500$
- $0 \leq v_i < N$ pentru fiecare $0 \leq i \leq N-1$
- $1 \leq M \leq 1\ 000$
- Scorul pentru acest subtask va fi 7 dacă toate testele sunt rezolvate corect și 0 în caz contrar

Subtask 2 (21 puncte)

- $1 \leq N \leq 10\ 000$
- $1 \leq M \leq 20\ 000$
- Scorul fiecarui test este $\min(1, \frac{1}{2^{\frac{L}{N}-1-\log_2 N}})$. Scorul subtask-ului este minimul dintre scorurile fiecarui test inmultit cu 21.

Subtask 3 (72 puncte)

- $1 \leq N \leq 40\ 000$
- $1 \leq M \leq 80\ 000$
- Scorul fiecarui test este $\min(1, \frac{1}{2^{\frac{L}{N}-2}})$. Scorul subtask-ului este minimul dintre scorurile fiecarui test inmultit cu 72.

Exemplu de interacțiune

La prima interacţiune, funcţia concurentului va fi apelată:

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

La rândul său, această funcție ar putea alege să salveze următorii biți pe dischetă:

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

Prima instanță a programului concurentului va fi terminată, prin urmare orice date stocate în memorie de acest program vor fi pierdute.

La a doua interacţiune, funcţia concurentului va fi apelată:

```
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});
```

Această funcţie ar trebui să returneze un şir de M întregi:

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