Problem Floppy

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

Roxette talált egy érdekes tömböt, amely N **különböző** egész számot tartalmaz: $v_0, v_1, v_2, \ldots, v_{N-1}$. El akarja menteni a floppy lemezére. Kevés szabad helye van, ezért nem tudja az egész tömböt elmenteni. Helyette azt tervezi, hogy csak egy bitekből álló sorozatot ment el, ami alapján az alábbi típusból tetszőleges kérdést meg tud válaszolni:

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

Más szóval minden kérdésre meg kell adni az adott intervallumban levő legnagyobb elem indexét.

Roxette most a segítségedet kéri. Kétszer is! Először megadja neked az érdekes tömböt, és neked meg kell adnod, hogy milyen bitsorozatot mentsen el a floppy lemezére. Másodszor, amikor néhány kérdésre válaszolnia kell, megadja neked a kérdéseket és a bitsorozatot, amit te mondtál neki, hogy mentsen a floppyra, neked pedig minden kérdésre meg kell adnod a helyes választ.

Interakció

Ez egy két interakciós feladat!
Két függvényt kell megvalósítanod. Az első közülük az alábbi:

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

Ez a függvény **pontosan egyszer** lesz meghívva az első interakcióban, és paraméterként megadja Roxette érdekes tömbjét. A függvény belsejében az alábbi függvényt kell meghívnod **pontosan egyszer** (amely az értékelőprogramban van implementálva):

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

Ezzel a függvénnyel megadod Roxette-nek, hogy milyen bitsorozatot mentsen a floppy lemezére. A `bits` paraméter egy karaktersorozat legyen (amelyben csak a `'0'` és `'1'` karakterek megengedettek). Az `L` paraméter jelzi a bitek számát (csak a C verzióban!).

A második függvény, amit meg kell valósítanod, az alábbi:

```
(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);
```

Ez a függvény **pontosan egyszer** lesz meghívva, a második interakcióban. Megadja Roxette érdekes tömbjének a hosszát (`N`), a bitsorozatot (`bits`), amit korábban Roxette (a te javaslatodra) elmentett a floppyjára, és M kérdést.

Az `i`-edik kérdés paraméterei `a[i]` és `b[i]`.

A függvénynek egy M egész számból álló tömböt kell visszaadnia, amely a válaszokat tartalmazza az egyes kérdésekre (az `i`-edik szám az `i`-edik kérdésre adott válasz legyen).

Fontos: A programod kétszer lesz lefuttatva, a két interakcióra külön. Tehát bármely adat, amit az első futtatás során számítasz ki, nem lesz elérhető a második futtatás alatt.

Korlátok

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

1. részfeladat (7 pont)

- $1 \leq N \leq 500$
- $0 \leq v_i < N$ for all $0 \leq i \leq N-1$
- $1 \leq M \leq 1\ 000$
- A részfeladatra kapott pontszám 7, ha minden tesztre helyes a válasz, egyébként 0.

2. részfeladat (21 pont)

- $1 \leq N \leq 10\ 000$
- $1 \leq M \leq 20\ 000$
- Minden tesztre a pontszám $\min(1, \frac{1}{2^{\frac{L}{N}-1-\log_2 N}})$. A részfeladatra kapott pontszám az egyes tesztekre kapott pontszámok minimuma 21-gyel szorozva.

3. részfeladat (72 pont)

- $1 \leq N \leq 40\ 000$
- $1 \leq M \leq 80\ 000$
- Minden tesztre a pontszám $\min(1, \frac{1}{2^{\frac{L}{N}-2}})$. A részfeladatra kapott pontszám az egyes tesztekre kapott pontszámok minimuma 72-vel szorozva.

Példa interakció

Az első interakcióban az alábbi függvényhívás történik:

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

Ebben a függvényben például az alábbi bitsorozatot adhatod meg a floppy-ra mentéshez:

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

Ekkor a programod első futtatásának vége lesz, tehát az összes memóriában tárolt adat elveszik.

A második interakcióban az alábbi függvényhívás történik:

```
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});
```

Ennek a függvénynek egy M elemű tömböt kell visszaadnia, a helyes válasz az alábbi:

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