Problem Cool Rotation

C header	`coolrot.h`
C++ header	`coolrot.h`

Little Square a decis sa cumpere un tort pentru Little Triangle, de la Brutaria InfO(1). Acest tort este foarte ciudat construit: este format din N felii, fiecare felie avand deasupra un anumit numar de bucati de ciocolata. Numerele bucatilor de ciocolata de pe felii sunt numere intregi distincte cuprinse intre 0 si $N-1$. Gusturile lui Little Triangle in ceea ce priveste torturile sunt destul de *excentrice*: gradul in care agreeaza un tort este determinat de numarul de perechi de felii (posibil neadiacente) pentru care felia din stanga are mai multe bucati de ciocolata decat felia din dreapta. Prin urmare, un tort cu mai putine astfel de perechi de felii ii va placea mai mult decat unul cu mai multe astfel de perechi. In cazul in care li se prezinta doua torturi similare conform acestui criteriu, ei il vor prefera pe cel mai mic in ordine lexicografica.

Little Square vrea ca lui Little Triangle sa ii placa cat mai mult posibil tortul pe care i-l cumpara. Pentru aceasta ei pot utiliza niste operatii, pe care brutaria le poate executa. Brutaria InfO(1) este foarte intelegatoare si nu tine cont de numarul de operatii executate. Fiecare operatie este determinata de doua numere intregi, d si x, unde d este un divizor al lui N si $0 < x < \frac{N}{d}$. Operatia consta impartirea tortului in blocuri de cate d felii si mutarea primelor x blocuri la finalul tortului. In ceea ce urmeaza sunt prezentate cateva exemple de operatii:

$$[0,1,2,3,4,5] \xrightarrow[d=2]{x=1} [2,3,4,5,0,1]$$
$$[0,1,2,3,4,5] \xrightarrow[d=3]{x=1} [3,4,5,0,1,2]$$
$$[0,1,2,3,4,5] \xrightarrow[d=1]{x=3} [3,4,5,0,1,2]$$

Din pacate, depinzand de zi, doar anumite tipuri de operatii sunt permise; pentru fiecare zi, doar anumite valori ale lui d sunt admise. Poti tu sa il ajuti pe Little Square sa execute operatii astfel incat sa obtina cel mai bun tort conform criteriului lui Little Triangle, in fiecare dintre cele Q zile care ii intereseaza?

Interaction protocol

Concurentul trebuie sa implementeze doua functii, `init` si `query`, cu urmatorul prototip.

```
void init(int n, const int a[], int q);
```

```
void query(int m, const int ds[]);
```

Concurentul poate sa utilizeze urmatoarea functie.

```
void update(int d, int x);
```

Functia `init` va fi apelata exact o singura data, inainte de orice apel al functiei `query`. Functia va primi `n`, lungimea tortului, `a`, un tablou continand numarul de bucati de ciocolata de pe fiecare felie (indexat de la 0), si `q`, numarul de zile care ne intereseaza.

Functia `query`, apelata de exact `q` ori, primeste `m`, numarul de valori d permise in ziua respectiva, si `ds`, un tablou ordonat (indexat de la 0) continand `m` divizori distincti ai lui n care reprezinta valorile

pentru d permise in ziua respectiva. Utilizand functia `update` (care corespunde unei operatii executate de Brutaria InfO(1)), feliile de tort trebuie sa fie plasate in aranjarea optimala in timpul apelului functiei `query`.

Toate apelurile functiei `query` sunt independente. Deci operatiile efectuate pe parcursul unui apel al functiei `query` nu sunt luate in considerare la urmatorul apel.

Evaluatorul oferit pentru testare citeste, de pe prima linie, N si Q, de pe a doua linie tabloul `a`, si descrierea celor Q interogari. Fiecare interogare este formata din doua linii. Pe prima linie a unei interogari se afla valoarea m pentru acea interogare (numarul de valori permise pentru d), si pe a doua linie valorile permise pentru d.

Pentru fiecare interogare, evaluatorul oferit pentru testare afiseaza operatiile executate utilizand `update`, fiecare pe o linie separata si la sfarsit secventa obtinuta.

Atentie! Concurentul nu trebuie sa implementeze functia main.

Restrictii

- $2 \leq N \leq 100.000$
- $1 \leq Q \leq 100.000$
- $1 \leq \sum m \leq 2.000.000$
- $1 \leq ds[i] \leq N$, pentru orice i, unde $0 \leq i < m$
- $ds[i]$ este un divizor al lui N, pentru orice i, unde $0 \leq i < m$
- Evaluatorul oferit pentru testare concurentilor nu este in mod obligatoriu acelasi cu cel utilizat pentru evaluare
- **Pentru fiecare update urmatoarele restrictii trebuie sa fie indeplinite:** $0 < x < \frac{N}{d}$

Subtask 1 (12 puncte)

- $N \leq 1.000$
- $Q = 1$
- $m = 1$

Subtask 2 (19 puncte)

- $m = 1$

Subtask 3 (11 puncte)

- $m = 2$

Subtask 4 (12 puncte)

- $m = 3$

Subtask 5 (11 puncte)

- $Q = 1$

Subtask 6 (35 puncte)

- Nu exista alte restrictii.

Exemple

input	output
6 2 1 3 4 0 2 5 2 3 2 1 2	3 1 0 2 5 1 3 4 1 3 4 0 2 5

Explicatie

In primul test, este optimal sa fie executata o operatie cu $d = 3, x = 1$, obtinandu-se secventa $0, 2, 5, 1, 3, 4$.
In al doilea test, este optimal sa nu fie executata nicio operatie, obtinandu-se secventa $1, 3, 4, 0, 2, 5$