

Classroom game

Ugledna srednja škola u Taškentu obilježava svoju godišnjicu organizovanjem igre između učenika i nastavnika. U učionici sjedi N učenika, numerisanih od 0 do $N - 1$. Svaki učenik drži komad papira na kojem se nalazi niz cijelih brojeva. Na početku je svaki papir prazan, pa sadrži prazan niz.

Učenici igraju igru sa M nastavnika, numerisanih od 0 do $M - 1$, i direktorom škole. Igra se odvija u M koraka, također numerisanih od 0 do $M - 1$. U koraku j , nastavnik j ulazi u učionicu i dešava se sljedeće:

- Neki učenici (možda nijedan) podignu ruku. Garantovano je da u svakom koraku barem jedan učenik **ne** podigne ruku, a svaki učenik može podići ruku **najviše jednom** tokom cijele igre.
- Nastavnik pogleda papire koje učenici trenutno drže i može **izmijeniti** papire učenika koji u ovom koraku **nisu** podigli ruku. Za svakog takvog učenika nastavnik može zamijeniti sadržaj njegovog papira novim (možda praznim) nizom cijelih brojeva. Svaki cijeli broj mora biti između 0 i 63, uključujući krajnje vrijednosti, a niz može imati najviše 63 elementa.
- Nakon tih izmjena, nastavnik j izlazi, a učenici razmjenjuju papire prema tajnoj permutaciji P . To jest, svaki učenik i ($0 \leq i < N$) daje svoj papir učeniku $P[i]$, pri čemu su $P[0], P[1], \dots, P[N - 1]$ N **različitih** cijelih brojeva između 0 i $N - 1$, uključujući krajnje vrijednosti. Vrijednosti P su **fiksne** tokom svih koraka igre, ali ih nastavnici i direktor škole ne znaju.

Kada se završi svih M koraka i obavi posljednja razmjena prema P , direktor škole ulazi u učionicu. Gledajući samo papire koje učenici drže, direktor škole mora za svakog učenika i ($0 \leq i < N$) odrediti broj koraka j u kojem je učenik i podigao ruku ili zaključiti da učenik i nikada nije podigao ruku.

Nastavnici ne mogu komunicirati s drugim nastavnicima ni s direktorom škole, osim putem nizova cijelih brojeva zapisanih na papirima. Svaki nastavnik zna u kojem koraku ulazi u učionicu.

Vaš zadatak je smisliti i implementirati strategiju za nastavnike i direktora škole koja ispravno određuje da li je i kada svaki učenik podigao ruku. Vaš broj bodova zavisit će od najveće dužine bilo kojeg niza zapisanog na papiru: manja najveća dužina donosi veći ili jednak broj bodova.

Detalji implementacije

Trebate implementirati dvije funkcija, jednu za nastavnike i jednu za direktora škole.

Funkcija koju trebate implementirati za **nastavnike** je:

```
std::vector<std::vector<int>> process_step(
    int N, int M, int R,
    std::vector<int> T,
    std::vector<std::vector<int>> A)
```

- N : broj učenika.
- M : broj koraka, a ujedno i broj nastavnika.
- R : broj trenutnog koraka (od 0 do $M - 1$).
- T : (možda prazan) niz koji se sastoji od indeksa učenika koji podignu ruku u ovom koraku, sortiranih u rastućem redoslijedu.
- A : niz dužine N koji opisuje papire, gdje je $A[i]$ ($0 \leq i < N$) niz cijelih brojeva na papiru koji učenik i drži na početku koraka.
- Ova funkcija se poziva tačno M puta po igri, redom za $R = 0, 1, \dots, M - 1$.

Funkcija treba vratiti niz B dužine N , koji predstavlja papire nakon nastavnikovih izmjena, u istom formatu kao A .

- Za svakog učenika i koji je podigao ruku (to jest, i se pojavljuje u T), papir se ne smije mijenjati, pa mora važiti $B[i] = A[i]$.
- Za svaki i takav da je $0 \leq i < N$, dužina niza $B[i]$ ne smije biti veća od 63, a svaki cijeli broj u $B[i]$ mora biti između 0 i 63, uključujući krajnje vrijednosti.

Funkcija koju trebate implementirati za **direktora škole** je:

```
std::vector<int> determine_steps(
    int N, int M, std::vector<std::vector<int>> A)
```

- N, M : isto kao iznad.
- A : niz dužine N , gdje je $A[i]$ niz cijelih brojeva na papiru koji učenik i drži nakon svih M koraka.
- Ova funkcija se poziva tačno jednom po igri, nakon posljednjeg poziva procedure `process_step`.

Funkcija mora vratiti niz D dužine N . Za svaki i takav da je $0 \leq i < N$ mora važiti:

- $D[i] = j$ ako je učenik i podigao ruku tokom koraka j , ili
- $D[i] = -1$ ako učenik i nikada nije podigao ruku tokom cijele igre.

Vaš program ne smije čuvati niti prenositi bilo kakve informacije iz jednog poziva funkcije `process_step` u drugi, osim putem povratne vrijednosti procedure `process_step`. Ako vaš program pokuša to uraditi, vaš broj bodova u CMS-u može biti netačan i može biti umanjen nakon završetka takmičenja. Imajte na umu da grader može istovremeno izvršavati više instanci vašeg

programa. To znači da se pozivi `process_step` i `determine_steps` iz iste igre mogu izvršavati u različitim instancama, a pozivi `process_step` i `determine_steps` iz različitih igara mogu izvršavati u istoj instanci. Svaki testni primjer sadrži najviše 5 igara.

Ograničenja

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- Svaki učenik podigne ruku **najviše jednom** tokom cijele igre. Drugim riječima, za svakog učenika i postoji najviše jedan korak j u kojem učenik i podigne ruku.
- U svakom koraku barem jedan učenik **ne** podigne ruku.

Podzadaci i bodovanje

Podzadatak	Bodovi	Dodatna ograničenja
1	4	$M = 1$
2	6	$N = 2$
3	9	$P[i] = i$ za sve $0 \leq i \leq N - 1$.
4	25	U svakom koraku najviše jedan učenik podigne ruku.
5	56	Nema dodatnih ograničenja.

Za svaki testni primjer vaš broj bodova je 0 (u CMS-u se prikazuje kao `Output isn't correct`) ako je povratna vrijednost bilo kojeg poziva funkcije `process_step` neispravna ili ako je povratna vrijednost bilo kojeg poziva funkcije `determine_steps` netačna.

U suprotnom, neka je C najveća dužina **bilo kojeg** niza u **bilo kojem** B koji vrati `process_step`. Tada se broj bodova za testni primjer u podzadatku koji vrijedi S bodova računa kao $S \cdot X$, gdje X zavisi od C prema sljedećoj tabeli:

Uslov	X
$C \leq 2$	1.00
$C = 3$	0.75
$C = 4$	0.55
$5 \leq C \leq 13$	$0.50 - 0.03 \cdot (C - 5)$
$14 \leq C \leq 63$	$0.19 \cdot \frac{64-C}{64-14} + 0.04$

Primjer

Razmotrimo slučaj sa $N = 4$ učenika, $M = 2$ nastavnika i permutacijom $P = [0, 3, 1, 2]$. Na početku su svi papiri prazni.

Grader prvo poziva:

```
process_step(4, 2, 0, [1], [[], [], [], []])
```

Učenik 1 je podigao ruku, pa se njegov papir ne smije mijenjati. Nastavnik 0 može odlučiti zapisati $[10]$ na papir učenika 0, zapisati $[1, 63, 4]$ na papir učenika 3 i ostaviti papir učenika 2 praznim. Da bi to uradila, funkcija mora vratiti $B = [[10], [], [], [1, 63, 4]]$.

Nakon što nastavnik 0 izađe, učenici razmjenjuju papire prema P . Nakon razmjene, papiri su $A = [[10], [], [1, 63, 4], []]$.

Grader sada poziva:

```
process_step(4, 2, 1, [0, 3], [[10], [], [1, 63, 4], []])
```

Učenici 0 i 3 su podigli ruku, pa se njihovi papiri ne smiju mijenjati. Nastavnik 1 može odlučiti zapisati $[0, 40]$ na papir učenika 1 i zapisati $[1, 50]$ na papir učenika 2. Da bi to uradila, funkcija mora vratiti $B = [[10], [0, 40], [1, 50], []]$.

Nakon što nastavnik 1 izađe, papiri se ponovo razmjenjuju prema P . Nakon razmjene, papiri su $A = [[10], [1, 50], [], [0, 40]]$.

Na kraju, grader poziva:

```
determine_steps(4, 2, [[10], [1, 50], [], [0, 40]])
```

Ova funkcija mora vratiti niz $D = [1, 0, -1, 1]$ jer su učenici 0 i 3 podigli ruku u koraku 1, učenik 1 je podigao ruku u koraku 0, a učenik 2 nikada nije podigao ruku. U ovom primjeru je $C = 3$.

Sample Grader

Format ulaza:

```
N M
Q[0] Q[1] ... Q[N-1]
P[0] P[1] ... P[N-1]
```

Ovdje je Q niz dužine N , gdje je $Q[i] = j$ ako učenik i podigne ruku tokom koraka j , ili $Q[i] = -1$ ako učenik i nikada ne podigne ruku tokom cijele igre.

Format izlaza:

Nakon svakog poziva funkcije `process_step`, sample grader ispisuje sadržaj papira. Neka je K dužina niza B , a $L[i]$ dužina niza $B[i]$ (za svaki $0 \leq i < K$). Sample grader ispisuje B u sljedećem formatu, **nakon čega slijedi prazan red**:

```
K
L[0] B[0][0] B[0][1] ... B[0][L[0]-1]
L[1] B[1][0] B[1][1] ... B[1][L[1]-1]
:
L[K-1] B[K-1][0] B[K-1][1] ... B[K-1][L[K-1]-1]
```

Sample grader zatim razmjenjuje papire prema permutaciji P . Ako je $K \neq N$, sample grader umjesto toga ispisuje poruku o grešci i završava izvršavanje.

Nakon poziva funkcije `determine_steps`, sample grader ispisuje:

```
C H
D[0] D[1] ... D[H-1]
```

Ovdje je H dužina niza D koji vrati `determine_steps`.