D. Népszámlálás (census)

Időlimit: 1 másodperc
Memória limit: 128 MiB

Egy kevésbé ismert tény Cesenatico-ról, hogy itt él egy titkos társaság, amely N női informatikusból áll. Ez a társaság igazán titkos; egyetlen tag sem ismeri a többiek kilétét.

A tagok közötti egyetlen kommunikáció csak közvetetten, a város különböző pontjain krétával felírt számokon keresztül zajlik. Százévente a társaság népszámlálást tart, hogy megszámolja a tagjait. A népszámlálás befejezése után minden tagnak tudnia kell a társaság teljes létszámát.

A népszámlálás több napon át tart. Minden nap az adott napon résztvevő tagok pontosan egy műveletet választanak és hajtanak végre: **olvasás**, **írás**, vagy a folyamatban való részvétel **abbahagyása**.

- Ha egy tag az **olvasás** mellett dönt, kiválaszt egy P helyet. Napközben ellátogat a P helyre, és elolvassa az ott felírt számot.
- Ha egy tag az **írás** mellett dönt, kiválaszt egy P helyet és egy V számot. Késő este ellátogat a P helyre, és az ott lévő számot V-re változtatja. Mivel már sötét van, nem tudja elolvasni a régi számot az új beírása előtt.
- Ha egy tag az **abbahagyás** mellett dönt, a következő napokban már semmilyen tevékenységet nem végez.

Ha egy tag látja, amint egy másik számot ír, megtudhatná annak kilétét. Ezért szigorúan tilos, hogy két vagy több tag ugyanazon a napon ugyanarra a helyre írjon. (Az olvasásra nincs ilyen korlátozás, mivel az diszkréten is elvégezhető.)

Ha egy vagy több tag olyan helyről olvas, ahová egy másik tag írni szeretne ugyanazon a napon, minden olvasás az írás előtt történik.

Hogyan tervezze meg a társaság a népszámlálási folyamatot, hogy minimalizálják a napok számát, amíg mindenki meg nem tudja a helyes taglétszámot?

Implementáció

⇨ Ez egy interaktív feladat, amelyben a programod ismeretlen számú példánya ($1 \leq N \leq 100$) fut egyszerre. Minden példány a társaság egy tagját szimulálja.

$10^{\{18\}}$ helyszín van. Egy hely P azonosítójának teljesítenie kell a $0 \leq P < 10^{\{18\}}$ feltételt. Kezdetben minden helyen $V = 0$ érték van írva.

Egy helyre beírt új V értéknek mindig egész számnak kell lennie, amelyre $0 \leq V \leq 10^9$ igaz. A legtöbb részfeladatban a V csak 0 vagy 1 lehet. Lásd a Pontozás részt a további részletekért.

Amikor a programod egy példánya elindul, először olvasson be egy sort két egész számmal, I és M $(0 \leq I \leq M - 1)$: a példány által képviselt tag egyedi azonosítója és a lehetséges azonosítók száma. Minden tesztesetnél minden példány ugyanazt az M értéket és különböző I értékeket kapja. Megjegyzés: lehetnek olyan azonosítók, amelyek nincsenek egyik taghoz sem rendelve.

Ezután a népszámlálási folyamat minden napjára a programodnak ki kell választania a végrehajtandó műveletet, és ennek megfelelően ki kell írnia egy sort:

Művelet	A művelet jelentése
`r` P	**Olvasás** a P helyen. A sor kiírása után a programodnak be kell olvasnia a P helyen lévő aktuális értéket.
`w` P V	**Írás** a P helyre a V új értékkel. Ha több példány ugyanazon a napon ugyanarra a P helyre ír, *Nem helyes* értékelést kapsz. A példákon és a 3. részfeladaton kívül $0 \leq V \leq 1$ értéket kell írnod; lásd a Pontozás részt.
`!` N	**Válasz és kilépés**: Jelentsd be, hogy N tag van, és hagyd abba a részvételt a népszámlálásban. A válaszadás után **a programodnak normálisan ki kell lépnie**. (Megjegyzés: a programod többi példánya még futhat további napokon, mielőtt válaszol és kilép.)

Ha a programod bármely példánya rossz N értéket ad meg, vagy megsérti a protokollt, vagy több mint 500 napot használ, vagy túllépi a (folyamatonkénti) idő/memória korlátot, a beküldésed *Nem helyes* értékelést kap az adott tesztesetre.

Ellenkező esetben a programod *(Részben) Helyes* értékelést kap a tesztesetre, és a D érték alapján lesz pontozva, ami a maximális napok száma, amíg valamely példány válaszolt. A teljes pontszámhoz minden tesztesetet $D \leq 61$ és $V \leq 1$ értékekkel kell megoldanod. Lásd a Pontozás részt a részletekért.

Kimenet ürítése (Flushing). Ha nem a sablonokat használod, ügyelj arra, hogy minden sor kiírása után ürítsd a standard kimenetet (flush), különben a programod *Nem helyes* értékelést kaphat. Pythonban ez automatikusan megtörténik, ha `input()`-ot használsz olvasáshoz. C++-ban a `cout << endl;` a soremelés mellett üríti a kimenetet; `printf` használata esetén használd az `fflush(stdout)` parancsot.

Korlátok

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Legfeljebb 500 napot használhatsz.

Pontozás

A programodat több, részfeladatokba csoportosított teszteseten teszteljük. Egy részfeladat pontszámának megszerzéséhez az összes benne lévő tesztesetet helyesen kell megoldanod.

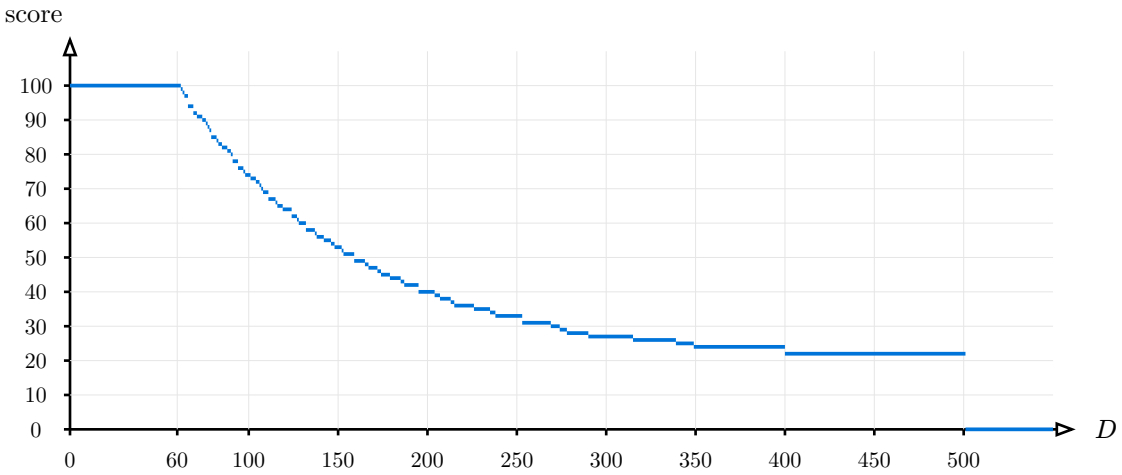
- **0. Részfeladat [0 pont]**: Példák (bármilyen egész számot írhatsz: $0 \leq V \leq 1\,000\,000\,000$).
- **1. Részfeladat [11 pont]**: $M \leq 100$, és az N tagnak $0, 1, ..., N-1$ azonosítói vannak.
- **2. Részfeladat [12 pont]**: $1 \leq N \leq 2$.
- **3. Részfeladat [22 pont]**: $M \leq 8000$, és bármilyen egész számot írhatsz: $0 \leq V \leq 1\,000\,000\,000$.
- **4. Részfeladat [55 pont]**: Nincsenek további megkötések.

Az 1., 2. és 4. részfeladatban minden írás **műveletnél csak $V = 0$ vagy $V = 1$ értéket írhatsz.**

Legyen M_s az s részfeladat maximális pontszáma (lásd fent), és D_s a napok legnagyobb száma, amelyet a programjaid egyike sem lép túl az s részfeladatban. Ekkor a score (a kapott pontszám):

$$\text{score}_s = \begin{cases} M_s & \text{, ha } D_s \leq 61 \\ M_s \cdot \left(0.2 + 0.8 \cdot 1.01^{(60-D_s)}\right) & \text{, ha } 61 < D_s \leq 500 \\ 0 & \text{, ha } 500 < D_s. \end{cases}$$

A score_s értékét részfeladatonként a legközelebbi egész számra kerekítjük, és a teljes pontszámod ezek összege. A teljes pontszámhoz minden teszteseten $D \leq 61$ és $V \leq 1$ feltételnek kell teljesülnie.



Ábra 1: Összpontszám, feltételezve, hogy minden részfeladatot ugyanazzal a maximális D-vel oldottunk meg.

Példák

Első példa. Minden oszloppár az értékelő és egy példány kommunikációját mutatja.

Értékelő	**Példány 0**
0 100	
	w 12 1
	r 50
1	
	! 5

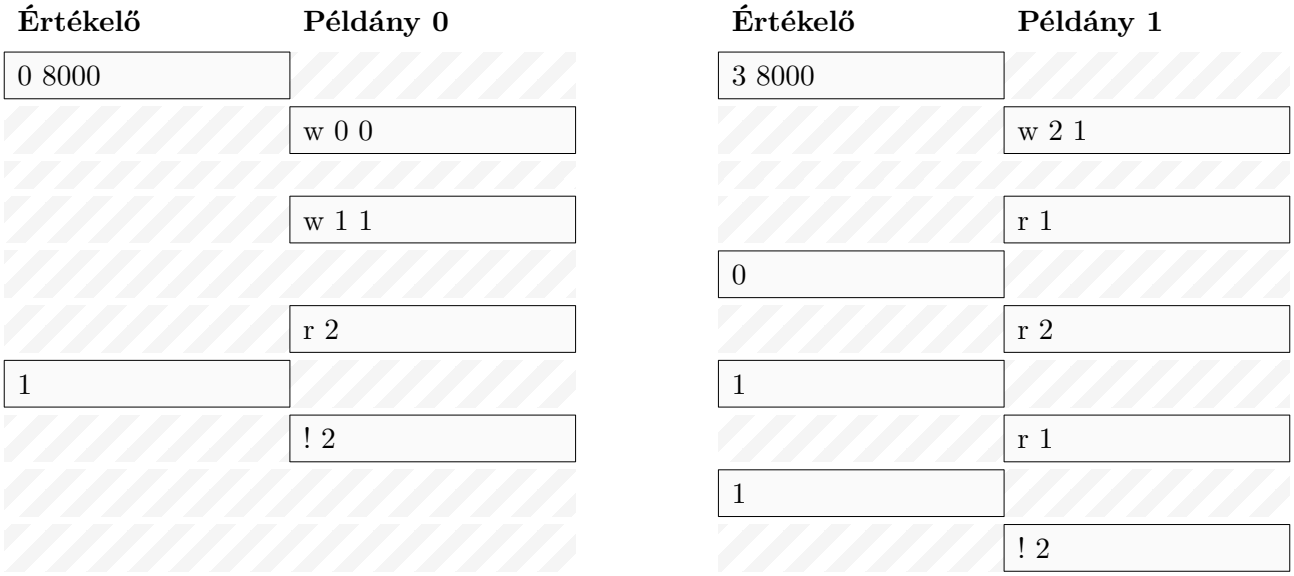
Értékelő	**Példány 1**
1 100	
	w 50 1
	r 7
1	
	r 1
1	
	! 5

Értékelő	**Példány 2**
2 100	
	w 99 0
	r 12
1	
	w 0 0
	! 5

Értékelő	**Példány 3**
3 100	
	w 7 1
	w 1 1
	! 5

Értékelő	**Példány 4**
4 100	
	r 5
0	
	! 5

Második példa.



Magyarázat

Első példa. $N = 5$ tagunk van egymást követő $0, 1, 2, 3, 4$ azonosítókkal és $M = 100$ (érvényes az 1., 3. és 4. részfeladatra). Az i. példány az i azonosítóval rendelkező tagnak felel meg. A fenti interakció csak egy lehetséges szabályos műveletsor, és **nem** egy hatékony vagy ésszerű stratégia; csak a protokoll működésének illusztrálására szolgál.

Második példa. $N = 2$ tagunk van, 0 és 3 azonosítóval, és $M = 8000$ (érvényes a 2., 3. és 4. részfeladatra). Az első napon a 0 azonosítójú tag 0-t ír a 0. helyre (nincs változás), a 3 azonosítójú tag pedig 1-et ír a 2. helyre.

location	0	1	2	3	4	...
value	0	0	1	0	0	...

A második napon a 0-s azonosítójú tag 1-et ír az 1. helyre, a 3-as azonosítójú tag pedig elolvassa ugyanezt a helyet. Figyeld meg, hogy az olvasás napközben történik, az esti írás előtt. Ezért a 3-as azonosítójú tag még 0-t lát.

location	0	1	2	3	4	...
value	0	1	1	0	0	...

A harmadik napon mindketten elolvassák a 2. helyet, ahová az 1-es van írva.

A negyedik napon a 0-s azonosítójú tag válaszol, hogy 2 tag van (helyes), miközben a 3-as azonosítójú tag elolvassa az 1-es értéket az 1. helyen. A 0-s azonosítójú tag ezután azonnal kilép, és a következő napokban már nem vesz részt a népszámlálási eljárásban.

Végül a $D = 5$. napon a megmaradt tag is helyesen válaszol, hogy $N = 2$.

Tesztelés

A megoldásod teszteléséhez egy egyszerű eszközt biztosítunk, amelyet a CMS-ből tölthetsz le. Az eszköz használata opcionális. Megjegyzés: a hivatalos CMS-beli ellenőrző (grader) különbözik a tesztelő eszköztől.

Az eszköz használatához egy bemeneti fájlra van szükség. Használhatod a biztosított példa bemeneteket: `census.input0.txt` és `census.input1.txt`, vagy készíthetsz sajátot is. A bemeneti fájlnak a tagok számával (N) és a lehetséges azonosítók számával (M) kell kezdődnie, amelyet egy sor követ N darab számmal, megadva a társaság tagjainak azonosítóit.

Python program esetén (általában `pypy3 census.py` paranccsal futtatva), a következő paranccsal futtasd a tesztelő eszközt:

```
python3 testing_tool.py pypy3 census.py < census.input0.txt
```

C++ program esetén először fordítsd le a megoldásodat a következő utasítással:

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o census census.cpp
```

majd futtasd a tesztelő eszközt:

```
python3 testing_tool.py ./census < census.input0.txt
```

Fontos megjegyezni, hogy ebben a feladatban a szabványos kimenetet (standard output) használják a kiértékelővel (graderrel) való kommunikációra, ezért nem szabad hibakeresésre használni. Ehelyett a szabványos hibakimenetet (standard error output, stderr) használhatod hibakeresésre. C++ nyelven használhatod például a következő parancsot: cerr << msg << endl;. Pythonban: print(msg, file=sys.stderr).

A tesztelő eszköz beolvassa és megjeleníti ezeket a stderr üzeneteket az összes program-példányod által végrehajtott lekérdezésekkel együtt. Fontos, hogy technikai okokból ezek az üzenetek kissé elcsúszva, nem teljesen szinkronban jelenhetnek meg egymáshoz képest.