D. Folketelling (census)

Tidsbegrensning: 1 sekund(er)
Minnebegrensning: 128 MiB

Et lite kjent faktum om Cesenatico er at det er basen til en hemmelig organisasjon med N kvinnelige informatikere. Denne organisasjonen er veldig hemmelig; ingen av medlemmene kjenner hverandre. Hvert medlem har en unik ID: et ikke-negativt heltall I.

Den eneste kommunikasjonen mellom medlemmene er indirekte, via tall tegnet med kritt på forskjellige steder i byen. Hvert 100. år gjennomfører organisasjonen en folketelling for å telle medlemmene sine. Etter at folketellingen er ferdig, skal alle medlemmene vite det totale antallet medlemmer i organsiasjonen.

Folketellingen foregår over flere dager. Hver dag vil hvert medlem som fortsatt er med i prosessen, velge og utføre nøyaktig én handling: å **lese**, å **skrive**, eller å **stoppe** å delta.

- Hvis et medlem velger å **lese**, velger hun et sted P. På dagtid drar hun til stedet P og leser tallet som står der.
- Hvis et medlem velger å **skrive**, velger hun et sted P og et tall V. Sent på kvelden besøker hun stedet P og endrer tallet som stod der til V. Siden det allerede er mørkt, kan hun ikke lese det gamle tallet før hun skriver det nye.
- Hvis et medlem velger å **stoppe**, gjør hun ikke noen flere handlinger de neste dagene.

Hvis ett medlem ser at et annet medlem skriver et tall, kunne hun kanskje kjent henne igjen. Derfor er det strengt forbudt for to eller flere medlemmer å velge å skrive på samme sted på samme dag. (Det er ingen slik begrensning for å lese, siden det kan gjøres diskré.)

Hvis ett eller flere medlemmer leser fra et sted hvor et annet medlem vil skrive på samme dag, skjer all lesing før det skrives.

Hvordan bør organisasjonen planlegge folketellingen for å minimere antall dager til alle vet det riktige antallet medlemmer?

Interaksjon

⇨ Dette er en interaktiv oppgave, hvor et ukjent antall instanser ($1 \leq N \leq 100$) av programmet ditt vil kjøre samtidig. Hver instans simulerer ett medlem av organisasjonen.

Det finnes 10^{18} steder. Tallet P for et sted må oppfylle $0 \leq P < 10^{18}$. Til å begynne med er verdien som er skrevet på alle steder $V = 0$.

Den nye verdien V som skrives på et sted må alltid være et heltall slik at $0 \leq V \leq 10^9$. I de fleste deloppgaver kan V bare være 0 eller 1. Se avsnittet om Poengsum for flere detaljer.

Når en instans av programmet ditt starter, skal den først lese en linje med to heltall, I og M ($0 \leq I \leq M - 1$): den unike ID-en til medlemmet av organisasjonen representert av denne instansen, og det totale antallet mulige ID-er. I hver test vil alle instanser få den samme verdien for M, og ulike verdier for I. Merk at det kan finnes ID-er som ikke er tildelt noe medlem.

Deretter, for hver dag i folketellingen, skal programmet ditt velge hvilken handling det vil utføre og skrive ut en linje basert på det:

Handling	Betydning
`r` P	**Les** sted P. Etter å ha printet denne linjen, skal programmet ditt lese inn en linje med den nåværende verdien som er skrevet på P.
`w` P V	**Skriv** på sted P den nye verdien V. Hvis flere instanser skriver på samme P på samme dag, får du bedømmelsen *Not correct*. Med unntak av eksemplene og deloppgave 3, må du skrive $0 \leq V \leq 1$; se avsnittet om Poengsum.
`!` N	**Svar og stopp**: meld fra om at det er N medlemmer og stopp å delta i folketellingen. Etter at du har svart, **må programmet ditt avslutte på vanlig måte**. (Merk at andre instanser av programmet ditt kan fortsette å kjøre i flere dager før de svarer og avslutter.)

Hvis en instans av programmet ditt svarer feil verdi for N, bryter protokollen, bruker mer enn 500 dager, eller overskrider tids-/minnegrensen (per instans), vil løsningen din bli bedømt som *Not correct* for den gitte testen.

Ellers vil programmet ditt bli bedømt *(Partially) Correct* på testen og få poeng basert på verdien D: det maksimale antallet dager noen av instansene brukte på å svare. For å få maksimal poengsum må du løse alle testcasene med $D \leq 61$ og $V \leq 1$. Se avsnittet om Poengsum for detaljer.

Flushing. Hvis du ikke bruker de vedlagte malene (templates), sørg for å flushe standard output etter å ha skrevet ut hver linje, ellers kan løsningen få *Not correct*. I Python skjer dette automatisk hvis du bruker `input()` for å lese linjer. I C++ flusher `cout << endl;` i tillegg til å skrive ut en ny linje; bruker du printf, bruk `fflush(stdout)`.

Begrensninger

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Du kan bruke maks 500 dager.

Poengsum

Løsningen din vil bli testet på et sett med deloppgaver (subtasks). Hver deloppgave inneholder et sett med tester. For å få poengene for en deloppgave må du løse alle testene i deloppgaven.

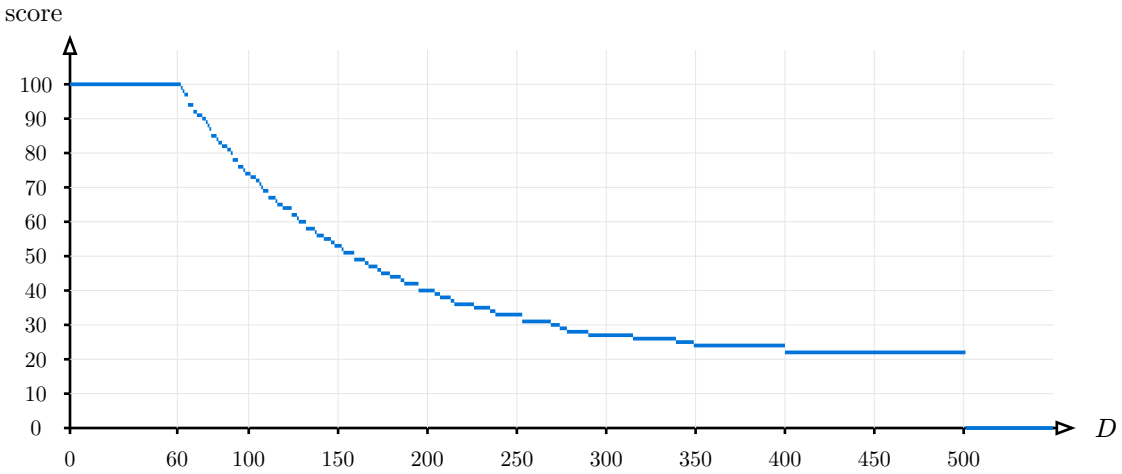
- **Deloppgave 0 [0 poeng]**: Eksempler (du kan skrive et hvilket som helst heltall $0 \leq V \leq 1\,000\,000\,000$).
- **Deloppgave 1 [11 poeng]**: $M \leq 100$, og de N medlemmene har ID-er $0, 1, ..., N-1$.
- **Deloppgave 2 [12 poeng]**: $1 \leq N \leq 2$.
- **Deloppgave 3 [22 poeng]**: $M \leq 8000$, og du kan skrive et hvilket som helst heltall $0 \leq V \leq 1\,000\,000\,000$.
- **Deloppgave 4 [55 poeng]**: Ingen ytterligere begrensninger.

I deloppgave 1, 2 og 4 kan du bare skrive $V = 0$ eller $V = 1$ i hver skrive-handling.

La X_s være maks antall poeng for deloppgave s (vist over), og D_s være det høyeste antallet dager noen av instansene dine bruker på en test i deloppgave s. Da blir:

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

Verdien av score_s blir rundet av til nærmeste heltall for hver deloppgave, og totalscoren din er summen av disse. For å få full poengsum for oppgaven, trenger du $D \leq 61$ og $V \leq 1$ på alle testene.



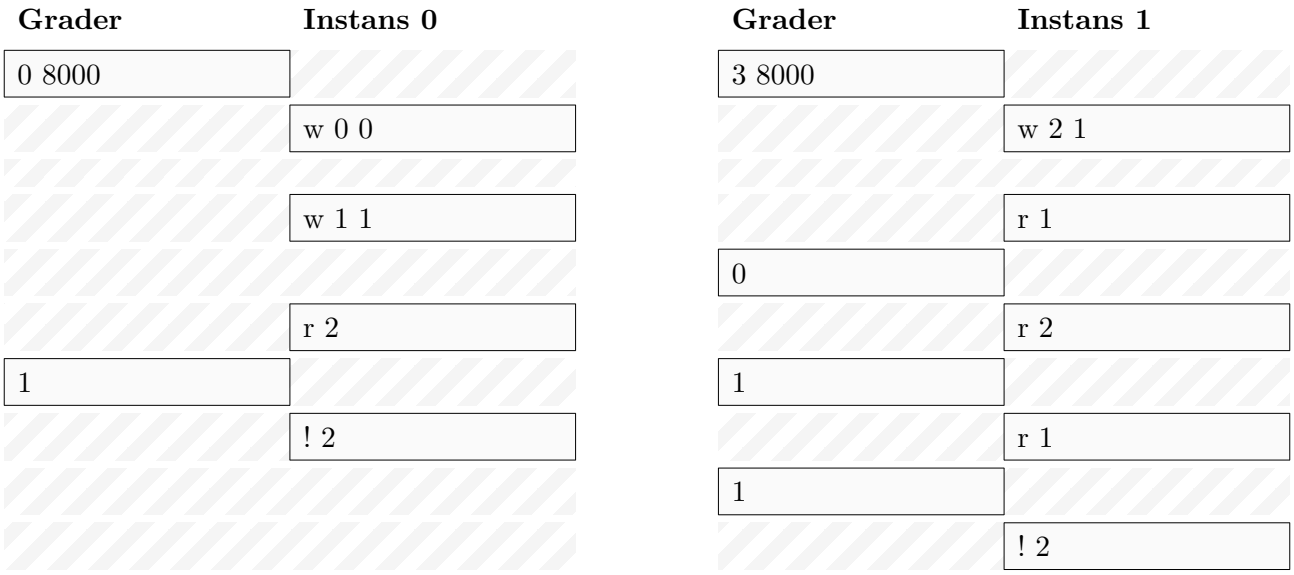
Figur 1: Totalscore, gitt at hver deloppgave blir løst med samme maksimum D.

Eksempler

Første eksempel. Hvert kolonnepar viser kommunikasjonen mellom graderen og en instans.

Gra.	Inst. 0	Gra.	Inst. 1	Gra.	Inst. 2	Gra.	Inst. 3	Gra.	Inst. 4
0 100		1 100		2 100		3 100		4 100	
	w 12 1		w 50 1		w 99 0		w 7 1		r 5
	r 50		r 7		r 12		w 1 1	0	
1		1		1					! 5
	! 5		r 1		w 0 0		! 5		
		1							
			! 5		! 5				

Andre eksempel.



Forklaring av eksempler

Første eksempel. Vi har $N = 5$ medlemmer med påfølgende ID-er $0, 1, 2, 3, 4$ og $M = 100$ (gyldig for deloppgave 1, 3 og 4). Instans i tilsvarer medlemmet med ID i. Interaksjonen over er bare én mulig gyldig sekvens med handlinger og er **ikke** ment å være en effektiv eller fornuftig strategi; den er kun vist for å illustrere hvordan protokollen fungerer.

Andre eksempel. Vi har $N = 2$ medlemmer, med ID-er 0 og 3, og $M = 8000$ (gyldig for deloppgave 2, 3 og 4). På den første dagen skriver medlemmet med ID 0 et 0-tall på sted 0 (ingen endring), og medlemmet med ID 3 skriver et 1-tall på sted 2.

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

På den andre dagen skriver ID 0 et 1-tall på sted 1, og ID 3 leser på det samme stedet. Merk at lesingen skjer på dagtid, før skrivingen skjer på kvelden. Dermed ser ID 3 fortsatt et 0-tall.

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

På den tredje dagen leser begge fra sted 2, hvor det står et 1-tall.

På den fjerde dagen svarer ID 0 at det er 2 medlemmer (som er riktig), mens ID 3 leser 1-tallet på lokasjon 1. ID 0 avslutter umiddelbart etter dette og deltar ikke de neste dagene.

Til slutt, på dag $D = 5$, svarer det gjenværende medlemmet også riktig at $N = 2$.

Testing

For å gjøre det enklere å teste løsningen din, tilbyr vi et enkelt verktøy du kan laste ned fra CMS. Verktøyet er valgfritt å bruke. Merk at den offisielle graderen på CMS er annerledes enn testverktøyet.

For å bruke verktøyet trenger du en inputfil. Du kan bruke de vedlagte eksempel-inputene `census.input0.txt` og `census.input1.txt`, eller lage din egen. Inputfilen skal starte med antall medlemmer N og mulige ID-er M, etterfulgt av en linje med N tall som spesifiserer ID-ene til medlemmene i organisasjonen.

For Python-programmer, f.eks. `census.py` (normalt kjørt som `pypy3 census.py`), kjør testverktøyet slik:

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

For C++-programmer må du først kompilere løsningen din:

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

og deretter kjøre testverktøyet:

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

Merk at i denne oppgaven brukes standard output til kommunikasjon med graderen, så det bør ikke brukes til debugging. I stedet kan du bruke standard error output (stderr). I C++ kan du bruke `cerr << msg << endl;`. I Python kan du bruke `print(msg, file=sys.stderr)`.

Testverktøyet vil lese og vise disse stderr-meldingene sammen med spørringene som utføres av alle programinstansene dine. Merk at grunnet tekniske årsaker kan de dukke opp litt usynkronisert med hverandre.