D. Censo (census)

Time limit: 1 seconds
Memory limit: 128 MiB

Un hecho poco conocido sobre Cesenatico es que es hogar de una sociedad secreta de N informáticas femeninas. Esta sociedad es muy secreta en verdad; ninguna de las miembros conoce a otra. Cada miembro tiene un ID único: un entero I tal que $0 \leq I \leq M-1$.

La única comunicación entre los miembros es indirecta, a través de números escritos con tiza en diferentes ubicaciones por toda la ciudad. Cada 100 años, la sociedad realiza un censo para contar a sus miembros. Después de completarse el censo, cada miembro debe saber el número total de miembros en la sociedad.

El censo se lleva a cabo durante varios días. Cada día, cada miembro que aún esté participando en el proceso elegirá y realizará exactamente una acción: *leer*, *escribir* o *dejar* de tomar parte en el censo.

- Si una miembro decide *leer*, escoge una locación P. Durante el dia, visita la locación P y lee el numero que este escrito ahí.
- Si una miembro decide *escribir*, escoge una locación P y un numero V. Tarde en la noche, visita la locacion P y cambia el número que estaba escrito por el número V. Debido a que ya es de noche y está oscuro, no puede leer el numero viejo antes de escribir el nuevo.
- Si una miembro decide *dejar* de tomar parte, ya no toma ninguna acción en los siguientes días.

Si una miembro ve a otra escribir un número, podría reconocerla. Por lo tanto, está estrictamente prohibido que dos o más miembros elijan escribir en la misma ubicación el mismo día. (No existe tal restricción para la lectura, ya que esta se puede hacer discretamente).

Si una o más miembros leen de una ubicación donde otra miembro quiere escribir el mismo día, todas las lecturas ocurren antes de la escritura.

¿Cómo debería la sociedad planificar su proceso de censo para minimizar el número de días hasta que todas sepan la cantidad correcta de miembros?

Implementation

> ⇨ Este es un problema interactivo, en el que un número desconocido de instancias ($1 \leq N \leq 100$) de tu programa se ejecutarán simultáneamente. Cada instancia simula una miembro de la sociedad.

Existen 10^{18} locaciones. El número P de una locación debe satisfacer $0 \leq P < 10^{18}$. Al principio, el valor escrito en todas las locaciones es $V = 0$.

El nuevo valor V escrito en una locación debe ser siempre un entero tal que $0 \leq V \leq 10^9$. En la mayoría de las subtareas, V solo puede ser 0 o 1. Revisa la sección de Scoring para más detalles.

Cuando una instancia de tu programa inicia, debe primero leer una línea con dos enteros, I y M ($0 \leq I \leq M-1$): el ID único de la miembro de la sociedad representado por esta instancia y el número total de IDs posibles. Dentro de cada caso de prueba, todas las instancias recibirán el mismo valor M y distintos valores I. Nota que puede haber IDs que no estén asignados a ninguna miembro.

Luego, para cada día en el proceso de censo, tu programa debe elegir la acción que desea realizar e imprimir una línea acorde a la acción a realizar:

Acción	Significado
`r` P	**Leer** locación P. Después de imprimir esta línea, tu programa debe leer una línea con el valor actual escrito en P.
`w` P V	**Escribir** en la locación P el nuevo valor V. Si múltiples instancias escriben en la misma P el mismo día, obtendrás el veredicto *Incorrecto*. Excepto para los ejemplos y la subtarea 3, debes escribir $0 \leq V \leq 1$; revisa la sección Scoring.
`!` N	**Responder y parar**: reportar que hay N miembros y parar de tomar parte en el censo. Después de responder, **tu programa debe salir normalmente**. (Nota que otras instancias de tu programa podrían continuar corriendo por días adicionales antes de responder y salir.)

Si alguna instancia de tu programa responde con un valor incorrecto de N, viola el protocolo, usa más de 500 días, o excede el límite de tiempo/memoria (por proceso), tu envio será juzgado como *Incorrecto* para el caso de prueba dado.

De otra forma, tu programa será *(Parcialmente) Correcto* en el caso de prueba y será puntuado basado en el valor D: el número máximo de días que alguna instancia tardó en responder. Para obtener la puntuación completa, necesitas resolver cada caso de prueba con $D \leq 61$ y $V \leq 1$. Revisa la sección de Scoring para más detalles.

Flushing. Si no estás usando las plantillas provistas, asegúrate de vaciar la salida estándar después de imprimir cada línea, o tu programa podría ser juzgado como *Incorrecto*. En Python, esto sucede automáticamente si usas `input()` para leer líneas. En C++, `cout << endl;` hace el flush además de imprimir un salto de línea; si usas printf, usa `fflush(stdout)`.

Constraints

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- ▸ Puedes usar como máximo 500 días.

Scoring

Tu programa será evaluado en varios casos de prueba agrupados en subtareas. Para obtener el puntaje de una subtarea, debes resolver correctamente todos los casos de prueba que contiene.

- **Subtask 0 [0 points]**: Ejemplos (puedes escribir cualquier entero $0 \leq V \leq 1\,000\,000\,000$).
- **Subtask 1 [11 points]**: $M \leq 100$, y los N miembros tienen IDs $0, 1, ..., N-1$.
- **Subtask 2 [12 points]**: $1 \leq N \leq 2$.
- **Subtask 3 [22 points]**: $M \leq 8000$, y puedes escribir cualquier entero $0 \leq V \leq 1\,000\,000\,000$.
- **Subtask 4 [55 points]**: Sin restricciones adicionales.

En las subtareas 1, 2 y 4, sólo puedes escribir $V = 0$ o $V = 1$ en cada acción de escritura.

Sea X_s el puntaje máximo para la subtarea s (mostrado arriba), y D_s el mayor número de días que alguna instancia de tu programa utilizó en un caso de prueba en la subtarea s. Entonces:

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

El valor de score_s se redondea al entero más cercano por subtarea, y el puntaje total es la suma de estos. Para obtener el puntaje completo para el problema, necesitas $D \leq 61$ y $V \leq 1$ en cada caso de prueba.

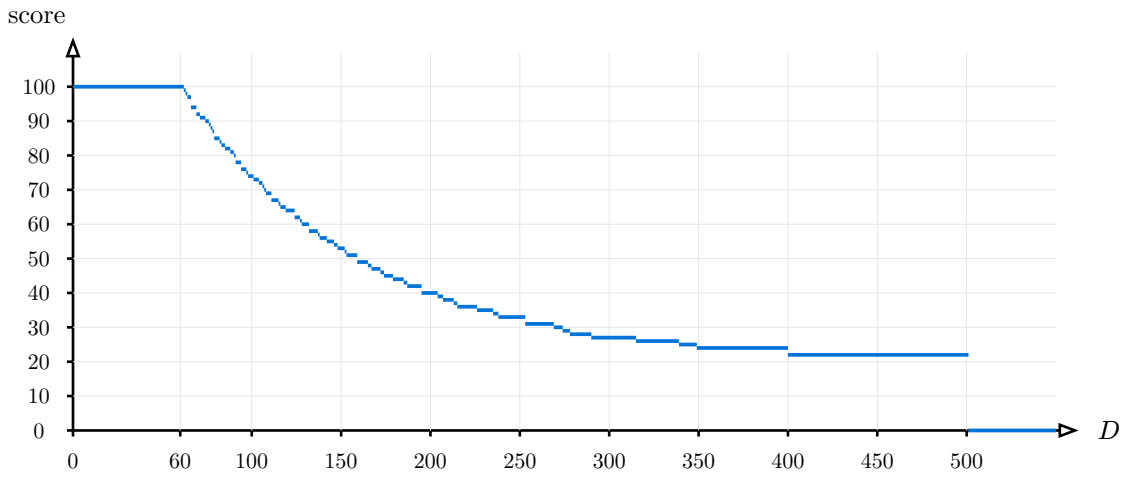


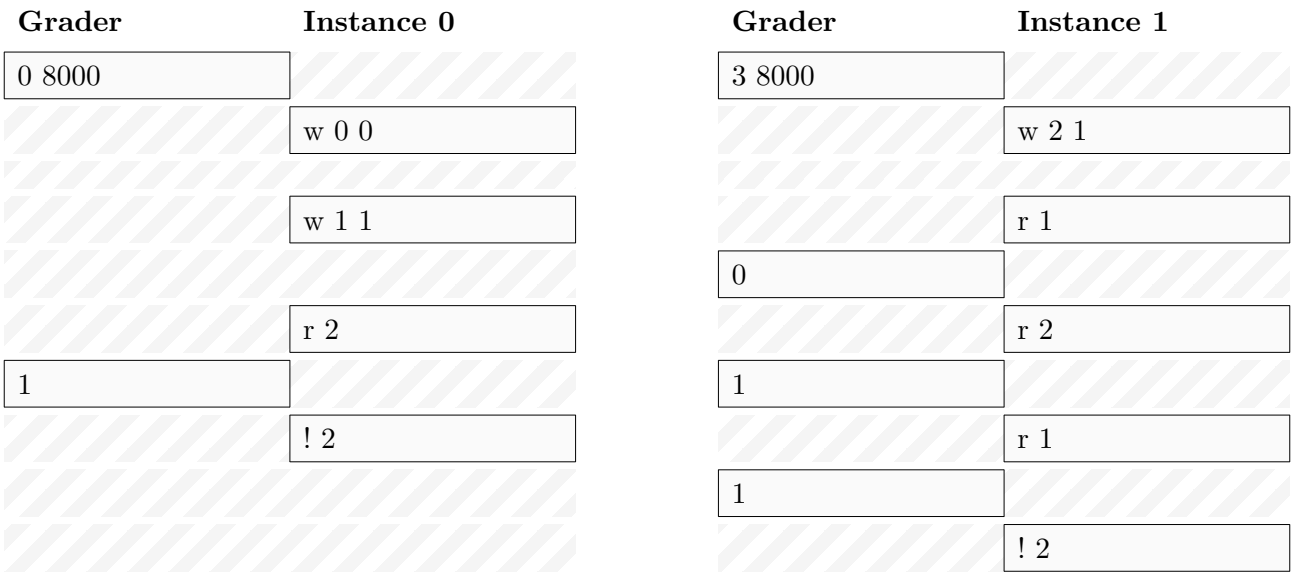
Figure 1: Puntaje total, asumiendo que cada subtarea es resuelta con el mismo máximo D.

Examples

Primer ejemplo. Cada par de columnas muestra la comunicación entre el grader y una instancia.

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
								0	
	r 50		r 7		r 12		w 1 1		! 5
1		1		1					
	! 5		r 1		w 0 0		! 5		
		1							
			! 5		! 5				

Segundo ejemplo.



Explanation

Primer Ejemplo. Tenemos $N = 5$ miembros con IDs consecutivos $0, 1, 2, 3, 4$ y $M = 100$ (válido para las subtareas 1, 3 y 4). La instancia i corresponde a la miembro con ID i. La interacción anterior es solo una secuencia legal posible de operaciones y **no** pretende ser una estrategia eficiente o sensata; se muestra solo para ilustrar cómo funciona el protocolo.

Segundo Ejemplo. Tenemos $N = 2$ miembros, con IDs 0 y 3, y $M = 8000$ (válido para las subtareas 2, 3 y 4). El primer día, la miembro con ID 0 escribe un 0 en la locación 0 (sin cambio), y la miembro con ID 3 escribe un 1 en la locación 2.

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

El segundo día, la miembro con ID 0 escribe un 1 en la locación 1, y la miembro con ID 3 lee esa misma locación. Nótese que la lectura ocurre durante el día, antes de la escritura en la tarde. Por lo tanto, la miembro con ID 3 sigue viendo un 0.

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

El tercer día, ambas leen la locación 2, donde hay un 1 escrito.

En el cuarto día, la miembro con ID 0 responde que hay 2 miembros (correcto), y la miembro con ID 3 lee el 1 en la locación 1. La miembro con ID 0 sale inmediatamente después de esto y no participa en los días siguientes.

Finalmente, en el día $D = 5$, la miembro restante también responde correctamente que hay $N = 2$ miembros.

Testing

Para facilitar el testeo de tu solución, proveemos una herramienta simple que puedes descargar desde CMS. La herramienta es opcional. Nota que el grader oficial en CMS es diferente de la herramienta de testing.

Para usar la herramienta necesitas un archivo de entrada. Puedes usar los archivos de entrada de ejemplo provistos `census.input0.txt` y `census.input1.txt`, o crear uno propio. El archivo de

entrada debe comenzar con el número de miembros N y el total de IDs posibles M, seguido de una línea con N números especificando los IDs de las miembros de la sociedad.

Para programas en Python, digamos `census.py` (normalmente ejecutado como `pypy3 census.py`) ejecuta la herramienta de testing de la siguiente manera:

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

Para programas en C++, primero compila tu solución:

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

y luego ejecuta la herramienta de testing:

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

Nota que en este problema la salida estándar se usa para la comunicación con el grader, por lo que no debe ser usada para debugear. En su lugar, puedes usar la salida de error estándar (stderr). En C++ puedes usar `cerr << msg << endl;`. En Python puedes usar `print(msg, file=sys.stderr)`.

La herramienta de testing leerá y presentará estos mensajes de stderr junto con las consultas realizadas por todas las instancias de tu programa. Nota que por razones técnicas, pueden aparecer ligeramente fuera de sincronía entre sí.