D. Recensement (census)

Limite de temps: 1 secondes
Limite de mémoire: 128 MiB

Un fait moins connu à propos de Cesenatico est que la ville abrite une société secrète de N informaticiennes. Cette société est en effet très secrète. Aucune informaticienne ne connaît l'identité d'une autre informaticienne. Chaque membre a un identifiant unique: un entier I non-négatif.

La seule façon de communiquer entre les membres est indirecte, en utilisant des nombres inscrits à la craie à divers emplacements de la ville. Tous les 100 ans, la société performe un recensement pour compter ses membres. Lorsque le recensement est terminé, chaque membre doit savoir le nombre total de membres dans la société.

Le recensement se déroule sur plusieurs jours. Chaque jour, les membres qui continuent de prendre part dans le processus choisissent d'effectuer une seule action : **lire**, **écrire** ou **arrêter** de prendre part au processus.

- Si une membre choisit de **lire**, elle choisit un emplacement P. Pendant la journée, elle visite alors l'emplacement P et lit le nombre qui y est écrit.
- Si une membre choisit d'**écrire**, elle choisit un emplacement P et un nombre V. Tard durant la nuit, elle visite alors l'emplacement P et change le nombre qui y était écrit à V. Puisqu'il fait nuit noire, elle ne peut pas lire l'ancien nombre avant d'écrire le nouveau.
- Si une membre choisit d'**arrêter**, elle ne prendra plus aucune action dans les jours qui suivent.

Si une membre voit une autre membre en train d'écrire un nombre, elle pourrait la reconnaître. Il est donc strictement interdit pour deux membres ou plus de choisir d'écrire au même emplacement le même jour. (Il n'y a pas de telle restriction pour lire, comme cela peut se faire discrètement.)

Si une des membres ou plus lisent à un emplacement où une autre membre souhaite écrire le même jour, toutes les lectures se font avant l'écriture.

Comment est-ce que la société secrète devrait planifier son processus de recensement pour minimiser le nombre de jours avant que tout le monde ne connaisse le nombre de membres dans la société.

Implémentation

⇨ Ce problème est un problème intéractif, dans lequel un nombre inconnu d'instances ($1 \leq N \leq 100$) de votre programme sont executées simultanément. Chaque instance simule un membre de la société.

Il y a 10^{18} emplacements. Le nombre P d'un emplacement doit satisfaire $0 \leq P < 10^{18}$. Initiallement, la valeur écrite à tous les emplacements est $V = 0$.

La nouvelle valeur V écrite à l'emplacement doit toujours être un entier tel que $0 \leq V \leq 10^9$. Dans la majorité des sous-tâches, V peut seulement être 0 ou 1. Pour plus de détails, lisez la section Score.

Quand une instance de votre programme se lance, vous devez lire sur une ligne deux entiers I et M ($0 \leq I \leq M - 1$): l'identifiant unique du membre de la société représenté par cette instance et le nombre total d'identifiants possibles. Pour chaque cas de test, toutes les instances auront la même

valeur de M et des valeurs de I distinces. Notez qu'il peut y avoir des identifiants qui ne sont pas assignés à une membre.

Ensuite, pour chaque jour dans le processus de recensement, votre programme doit choisir une action qu'il veut effectuer et afficher une ligne :

Action	Description
`r` P	**Lire** à l'emplacement P. Après avoir affiché cette ligne, votre programme doit lire sur une ligne un entier, la valeur actuelle écrite à l'emplacement P.
`w` P V	**Ecrire** à l'emplacement P la nouvelle valeur V. Si plusieurs instances écrivent au même P le même jour, vous recevrez le verdict *Not correct*. A part pour les exemples et la sous-tâche 3, vous devez écrire $0 \leq V \leq 1$; regardez la section Score pour plus de détails.
`!` N	**Arrêter**: afficher qu'il y a N membres et arrêter de prendre part au processus. Après avoir affiché cette ligne, **votre programme doit arrêter de s'exécuter normalement**. (Notez que d'autres instances de votre programme peuvent continuer de s'exécuter pour plusieurs jours avant de répondre et de s'arrêter.)

Si l'une des instances de votre programme affiche la mauvaise valeur de N, ne respecte pas le protocole, prend plus que 500 jours pour s'arrêter, ou dépasse la limite de temps ou de mémoire (par instance), votre soumission sera jugée comme *Not correct* pour ce cas de test.

Sinon, votre programme sera jugé *(Partially) Correct* sur le cas de test et le score sera basé sur la valeur D: le nombre maximum de jour qu'une de vos instances à pris à répondre et s'arrêter. Pour obtenir un score parfait, vous devez résoudre tous les cas de tests avec $D \leq 61$ et $V \leq 1$. Regardez la section Score pour plus de détails.

Flush. Si vous n'utilisez pas l'une des templates données, n'oubliez pas de flush la sortie standard après avoir affiché chaque ligne, sinon votre programme pourrait être jugé comme *Not correct*. En Python, cela a lieu automatiquement si vous utilisez `input()` pour lire des lignes. En C++, `cout << endl;` flush en plus d'afficher une nouvelle ligne. Si vous utilisez printf, utilisez `fflush(stdout)`.

Contraintes

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Vous pouvez prendre au plus 500 jours pour répondre.

Score

Votre programme sera testé sur plusieurs cas de tests groupés en plusieurs sous-tâches. Pour obtenir le score d'une sous-tâche, vous devez correctement résoudre tous les cas de tests contenus dans celle-ci.

- **Sous-tâche 0 [0 points]**: Exemples (vous pouvez écrire n'importe quel entier V tel que $0 \leq V \leq 1\,000\,000\,000$).
- **Sous-tâche 1 [11 points]**: $M \leq 100$, et les N membres ont pour identifiants $0, 1, ..., N-1$.
- **Sous-tâche 2 [12 points]**: $1 \leq N \leq 2$.
- **Sous-tâche 3 [22 points]**: $M \leq 8000$, et vous pouvez écrire n'importe quel entier V tel que $0 \leq V \leq 1\,000\,000\,000$.
- **Sous-tâche 4 [55 points]**: Aucune contrainte supplémentaire.

Dans les sous-tâches 1, 2, et 4, vous ne pouvez écrire que $V = 0$ ou $V = 1$ lors d'une opération d'écriture.

Soit X_s le nombre maximum de points pour la sous-tâche s (comme décrit ci-dessus), et D_s le plus grand nombre de jours que votre programme a pris pour s'arrêter sur un test de la sous-tâche s. Alors :

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

La valeur de score_s est arrondie à l'entier le plus proche par sous-tâche, et votre score total est la somme des valeurs des sous-tâches. Pour obtenir tous les points pour ce problème, vous devez respecter $D \leq 61$ et $V \leq 1$ pour chaque sous-tâche.

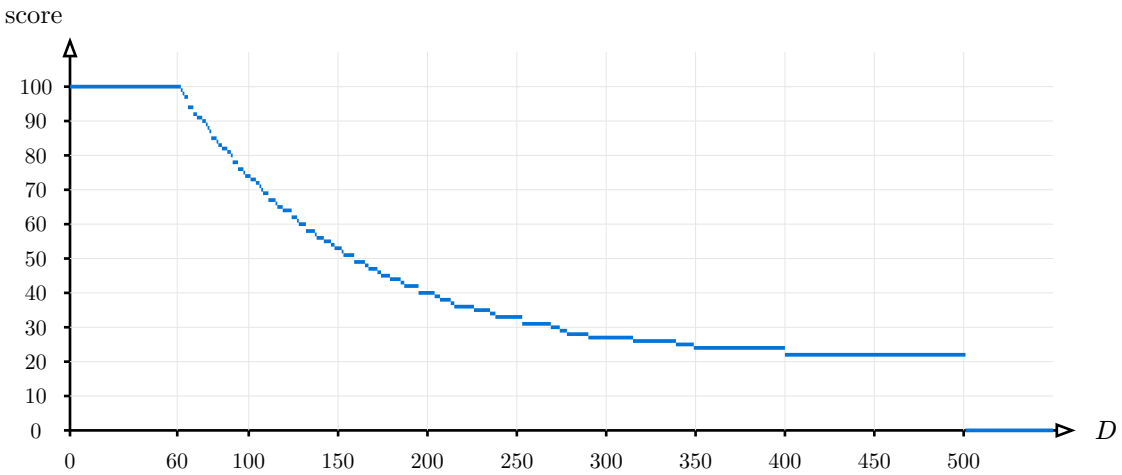

Fig. 1. – Score total, en supposant que chaque sous-tâche est résolue avec le même D maximum D.

Exemples

Premier exemple. Chaque paire de colonnes montre la communication entre le grader et une instance.

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				

Second exemple.

Grader	Instance 0
0 8000	
	w 0 0
	w 1 1
	r 2
1	
	! 2

Grader	Instance 1
3 8000	
	w 2 1
	r 1
0	
	r 2
1	
	r 1
1	
	! 2

Explication

Premier Exemple. Il y a $N = 5$ membres avec des identifiants consécutifs $0, 1, 2, 3, 4$ et $M = 100$ (valide pour les sous-tâches 1, 3 et 4). L'instance i correspond au membre avec l'ID i. L'interaction ci-dessus correspond à une séquence légale d'opérations et **n'est pas** supposée être une solution efficace ou sensée. Elle est seulement décrite pour illustrer comment fonctionne le protocole.

Deuxième Exemple. Il y a $N = 2$ membres, avec les identifiants 0 et 3, et $M = 8000$ (valide pour les sous-tâches 2, 3 et 4). Le premier jour, la membre avec l'identifiant 0 écrit la valeur 0 à l'emplacement 0 (aucun changement), et la membre avec l'identifiant 3 écrit 1 à l'emplacement 2.

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

Le second jour, l'identifiant 0 écrit la valeur 1 à l'emplacement 1, et l'identifiant 3 écrit au même emplacement. Notez que la lecture a lieu la journée, tandis que l'écriture a lieu pendant la soirée. Ainsi, l'identifiant 3 voit l'ancienne valeur 0.

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

Le troisième jour, elles lisent toutes les deux à l'emplacement 2, où la valuer 1 est écrite.

Le quatrième jour, l'identifiant 0 s'arrête et affiche qu'il y a 2 membres (ce qui est correct), tandis que l'identifiant 3 écrit la valeur 1 à l'emplacement 1. L'identifiant 0 s'arrête immédiatement après et ne participe plus dans les jours suivants.

Finalement, au jour $D = 5$, la membre restante s'arrête et affiche correctement que $N = 2$.

Outil de test

Pour faciliter l'évaluation de votre solution en local, nous vous donnons accès à un outil simple que vous pouvez télécharger sur CMS. Il est optionnel d'utiliser cet outil. Notez que le grader officiel sur CMS est différent de cet outil.

Pour utiliser l'outil, vous avez besoin d'un fichier d'entrée. Vous pouvez utiliser ceux des exemples `census.input0.txt` et `census.input1.txt`, ou faire vos propres tests.

Le fichier d'entrée doit contenir le nombre de membres N et le nombre d'identifiants possibles M, suivi par une ligne avec N nombres spécifiant les identifiants des membres de la société.

Pour un programme Python, par exemple `census.py` (lancé normalement avec `pypy3 census.py`), vous devez lancer l'outil de test avec :

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

Pour un programme C++, vous devez d'abord compiler votre solution avec :

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

et puis lancer le testing tool avec :

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

Notez que dans ce problème, la sortie standard est utilisée pour communiquer avec le grader, donc il ne peut pas être utilisé pour débugguer. A la place, vous pouvez utiliser la sortie standard d'erreur (stderr). En C++, vous pouvez utiliser `cerr << msg << endl;`. En python, vous pouvez utiliser `print(msg, file=sys.stderr)`.

L'outil de test va lire et afficher les messages sur la sortie standard d'erreur ainsi que les requêtes faites par toutes les instances de votre programme. Notez que pour des raisons techniques, les messages peuvent s'afficher dans un ordre légèrement différent les uns par rapport aux autres.