D. التعداد (census)

الحد الزمني: 1 ثواني
حد الذاكرة: 128MiB

من الحقائق غير المعروفة كثيراً عن Cesenatico هي أنها موطن لجمعية سرية مكونة من N عالمة حاسوب. هذه الجمعية سرية للغاية؛ فليس أي عضو يعرف الآخر. لكل عضوة معرف (ID) فريد: عدد صحيح غير سالب I.

وسيلة التواصل الوحيدة بين الأعضاء هي بطريقة غير مباشرة، عبر أرقام مكتوبة بالطباشير في مواقع مختلفة حول المدينة. كل 100 عام، تجري الجمعية تعداداً لإحصاء أعضائها. بعد اكتمال التعداد، يجب أن تعرف كل عضوة إجمالي عدد الأعضاء في الجمعية.

يستمر التعداد على مدى عدة أيام. في كل يوم، تختار كل عضوة لا تزال تشارك في العملية إجراءً واحداً فقط لتقوم به: **القراءة**، أو **الكتابة**، أو **التوقف** عن المشاركة.

- إذا اختارت العضوة **القراءة**، فهي تختار موقعاً P. خلال النهار، تزور الموقع P وتقرأ الرقم المكتوب هناك.
- إذا اختارت العضوة **الكتابة**، فهي تختار موقعاً P ورقمًا V. في وقت متأخر من المساء، تزور الموقع P وتغير الرقم الذي كان مكتوباً هناك إلى V. وبما أن الجو يكون مظلماً بالفعل، فلا يمكنها قراءة الرقم القديم قبل كتابة الرقم الجديد.
- إذا اختارت العضوة **التوقف**، فإنها لا تقوم بأي إجراءات أخرى في الأيام التالية.

إذا رأت عضوة عضوة أخرى تكتب رقماً، فقد تتعرف عليها. لذلك، يُمنع منعاً باتاً على عضوتين أو أكثر اختيار الكتابة في نفس الموقع في نفس اليوم. (لا يوجد مثل هذا القيد للقراءة، حيث يمكن القيام بذلك بتكتم.)

إذا قامت عضوة أو أكثر بالقراءة من موقع ترغب عضوة أخرى في الكتابة فيه في نفس اليوم، تحدث جميع عمليات القراءة قبل الكتابة.

كيف يمكن للجمعية التخطيط لعملية التعداد لتقليل عدد الأيام حتى يعرف الجميع العدد الصحيح للأعضاء؟

التنفيذ

⇨ هذه مسألة تفاعلية، حيث سيتم تنفيذ عدد غير معروف من النسخ $(1 \leq N \leq 100)$ من برنامجك في وقت واحد. تمثل كل نسخة عضواً واحداً في الجمعية.

هناك 10^{18} موقعاً. يجب أن يستوفي رقم الموقع P الشرط $0 \leq P < 10^{18}$. في البداية، تكون القيمة المكتوبة في جميع المواقع هي $V = 0$.

يجب أن تكون القيمة الجديدة V المكتوبة في موقع ما دائماً عدداً صحيحاً بحيث $0 \leq V \leq 10^9$. في معظم المهام الفرعية (subtasks)، لا يمكن أن تكون V إلا 0 أو 1. انظر قسم نظام النقاط لمزيد من التفاصيل.

عندما تبدأ نسخة من برنامجك، يجب عليها أولاً قراءة سطر يحتوي على عددين صحيحين، I و M $(0 \leq I \leq M - 1)$: المعرف الفريد لعضو الجمعية الذي تمثله هذه النسخة وإجمالي عدد المعرفات الممكنة. ضمن كل حالة اختبار، ستحصل جميع النسخ على نفس القيمة M وقيم متميزة لـ I. لاحظ أنه قد تكون هناك معرفات غير مسندة لأي عضو.

ثم، لكل يوم في عملية التعداد، يجب على برنامجك اختيار الإجراء الذي يريد القيام به وطباعة سطر بناءً على ذلك:

الإجراء	المعنى
`r` P	**قراءة** الموقع P. بعد طباعة هذا السطر، يجب أن يقرأ برنامجك سطراً يحتوي على القيمة الحالية المكتوبة في P.
`w` P V	**الكتابة** في الموقع P بالقيمة الجديدة V. إذا قامت نسخ متعددة بالكتابة في نفس الموقع P في نفس اليوم، فستحصل على حكم غير صحيح *(Not correct)*. باستثناء الأمثلة والمهمة الفرعية 3، يجب عليك كتابة $1 \leq V \leq 0$؛ راجع قسم نظام النقاط.

N !	**الإجابة والتوقف:** أبلغ عن وجود N عضواً وتوقف عن المشاركة في التعداد. بعد الإجابة، **يجب أن ينهي برنامجك عمله بشكل طبيعي.** (لاحظ أن النسخ الأخرى من برنامجك قد تستمر في العمل لأيام إضافية قبل أن تجيب وتنهي عملها.)

إذا أجابت أي نسخة من برنامجك بقيمة خاطئة لـ N، أو خالفت البروتوكول، أو استخدمت أكثر من 500 يوم، أو تجاوزت حد الوقت/الذاكرة (لكل عملية)، فسيتم الحكم على مشاركتك بـ غير صحيح لحالة الاختبار المحددة.

بخلاف ذلك، سيكون برنامجك صحيحاً *(جزئياً)* في حالة الاختبار وسيتم تقييمه بناءً على القيمة D: الحد الأقصى لعدد الأيام التي استغرقتها أي نسخة للإجابة. للحصول على الدرجة الكاملة، يجب عليك حل كل حالة اختبار بحيث $D \leq 61$ و $V \leq 1$. راجع قسم نظام النقاط لمزيد من التفاصيل.

تفريغ المخرجات (Flushing). إذا كنت لا تستخدم القوالب المقدمة، فتأكد من تفريغ المخرجات القياسية (flush standard output) بعد طباعة كل سطر، وإلا فقد يتم الحكم على برنامجك بـ غير صحيح. في لغة Python، يحدث هذا تلقائياً إذا استخدمت `input()` لقراءة الأسطر. في C++، يقوم `cout << endl;` بالتفريغ بالإضافة إلى طباعة سطر جديد؛ إذا كنت تستخدم printf، استخدم `fflush(stdout)`.

القيود

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- يمكنك استخدام 500 يوم كحد أقصى.

توزيع الدرجات

سيتم اختبار برنامجك على عدة حالات اختبار مجمعة في مهام فرعية. للحصول على درجة مهمة فرعية، يجب عليك حل جميع الاختبارات التي تحتوي عليها بشكل صحيح.

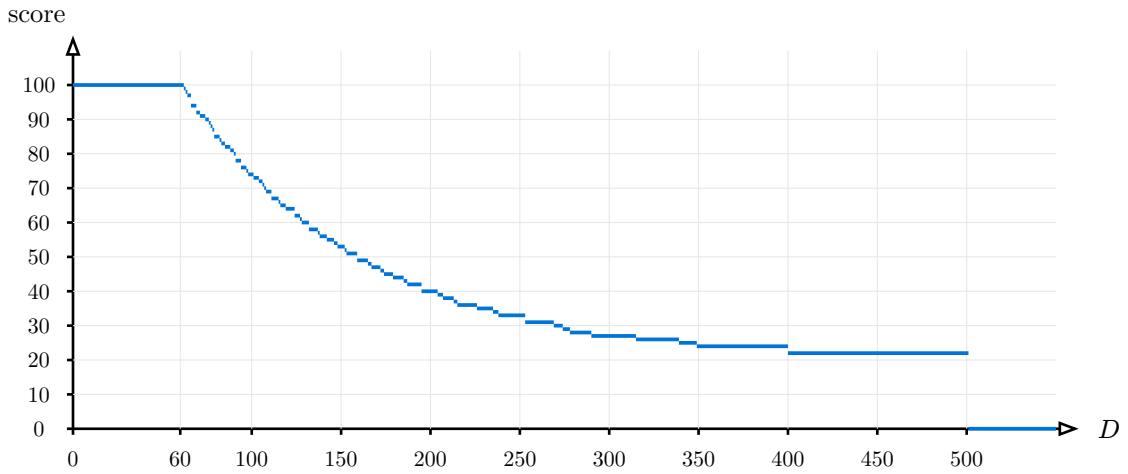
- المهمة الفرعية **0** [**0** نقاط]: الأمثلة (يمكنك كتابة أي عدد صحيح $0 \leq V \leq 1\,000\,000\,000$).
- المهمة الفرعية **1** [**11** نقاط]: $M \leq 100$، والأعضاء N لديهم معرفات $0, 1, ..., N-1$.
- المهمة الفرعية **2** [**12** نقاط]: $1 \leq N \leq 2$.
- المهمة الفرعية **3** [**22** نقاط]: $M \leq 8000$، ويمكنك كتابة أي عدد صحيح $0 \leq V \leq 1\,000\,000\,000$.
- المهمة الفرعية **4** [**55** نقاط]: لا توجد قيود إضافية.

في المهام الفرعية 1 و 2 و 4، يمكنك فقط كتابة $V = 0$ أو $V = 1$ في كل إجراء كتابة.

ليكن X_s هو الحد الأقصى للنقاط للمهمة الفرعية s (الموضحة أعلاه)، و D_s أكبر عدد من الأيام تستخدمه أي من برامجك في اختبار ضمن المهمة الفرعية s. إذن:

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

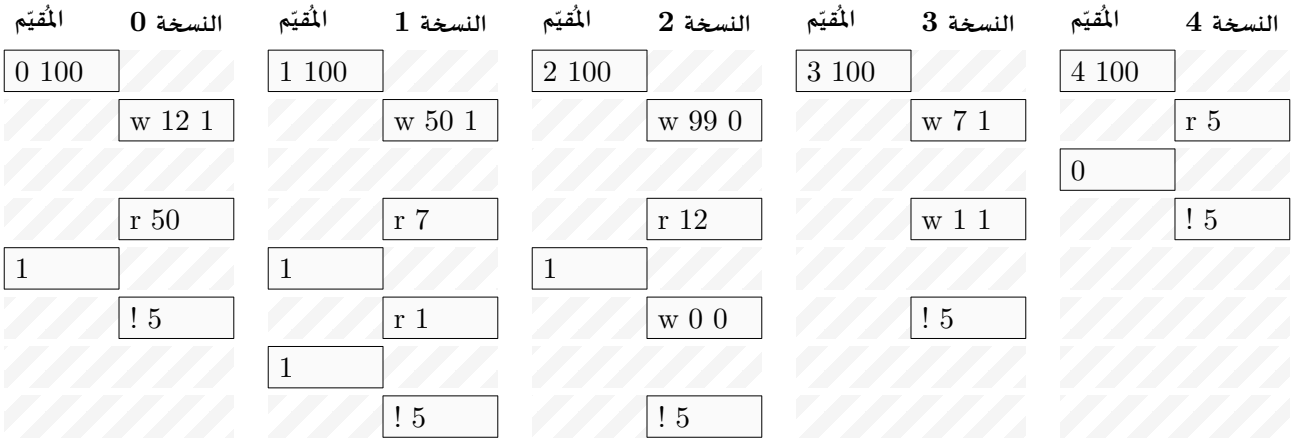
يتم تقريب قيمة score_s إلى أقرب عدد صحيح لكل مهمة فرعية، ودرجتك الإجمالية هي مجموع هذه القيم. للحصول على الدرجة الكاملة للمسألة، تحتاج إلى $D \leq 61$ و $V \leq 1$ في كل حالة اختبار.



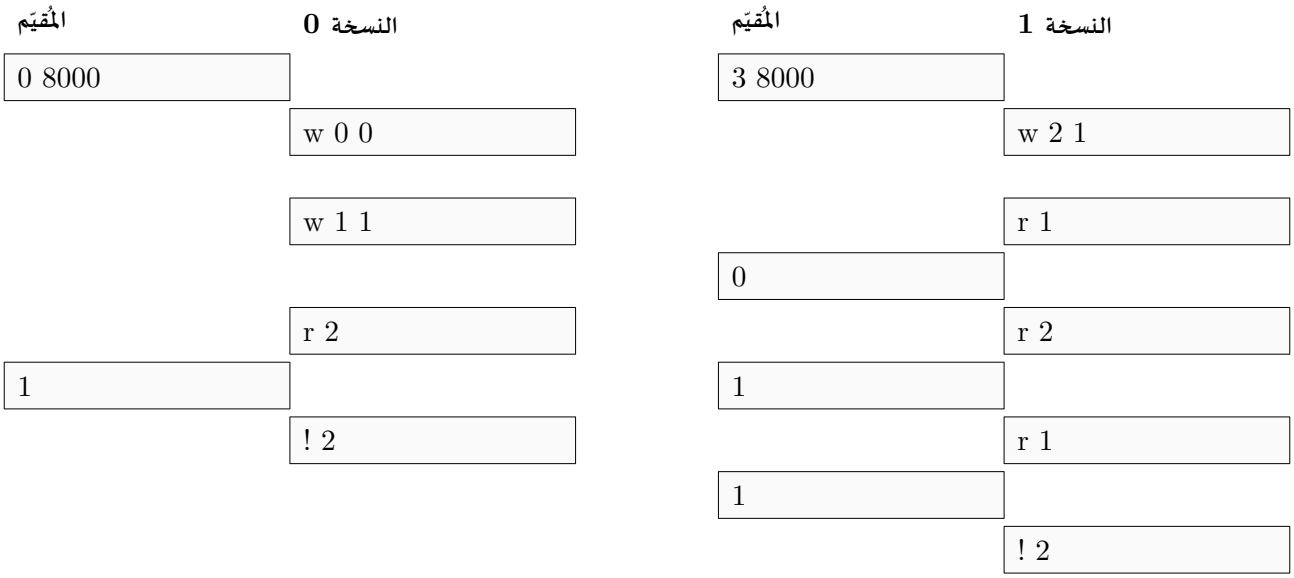
شكل 1: الدرجة الإجمالية، بافتراض حل كل مهمة فرعية بنفس الحد الأقصى لـ D.

Examples

المثال الأول. يظهر كل زوج من الأعمدة التواصل بين المُقيّم ونسخة واحدة.



المثال الثاني.



الشرح

المثال الأول. لدينا $N = 5$ أعضاء بمعرفات متتالية $0, 1, 2, 3, 4$ و $M = 100$ (صالحة للمهام الفرعية 1 و 3 و 4). النسخة i تتوافق مع العضو ذو المعرف i. التفاعل أعلاه هو مجرد تسلسل قانوني ممكن للعمليات **وليس** المقصود منه أن يكون استراتيجية فعالة أو منطقية؛ تم عرضه فقط لتوضيح كيفية عمل البروتوكول.

المثال الثاني. لدينا $N = 2$ أعضاء، بمعرفات 0 و 3، و $M = 8000$ (صالحة للمهام الفرعية 2 و 3 و 4). في اليوم الأول، يكتب العضو ذو المعرف 0 الرقم 0 في الموقع 0 (لا تغيير)، ويكتب العضو ذو المعرف 3 الرقم 1 في الموقع 2.

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

في اليوم الثاني، يكتب المعرف 0 الرقم 1 في الموقع 1، ويقرأ المعرف 3 نفس الموقع. لاحظ أن القراءة تحدث خلال النهار، قبل الكتابة في المساء. ومن ثم، لا يزال المعرف 3 يرى 0.

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

في اليوم الثالث، يقرآن كلاهما الموقع 2، حيث كُتب الرقم 1.

في اليوم الرابع، يجيب المعرف 0 بأن هناك 2 أعضاء (صحيح)، بينما يقرأ المعرف 3 الرقم 1 في الموقع 1. يخرج المعرف 0 فوراً بعد ذلك ولا يشارك في الأيام القادمة.

أخيراً، في اليوم $D = 5$، يجيب العضو المتبقي أيضاً بشكل صحيح بأن $N = 2$.

الاختبار

لتسهيل اختبار حلك، نوفر أداة بسيطة يمكنك تنزيلها من CMS. استخدام الأداة اختياري. لاحظ أن المُقيّم الرسمي على CMS يختلف عن أداة الاختبار.

لاستخدام الأداة، تحتاج إلى ملف إدخال. يمكنك استخدام أمثلة المدخلات المقدمة `census.input0.txt` و `census.input1.txt`، أو إنشاء ملف خاص بك. يجب أن يبدأ ملف الإدخال بعدد الأعضاء N والمعرفات الممكنة M، متبوعاً بسطر يحتوي على N أرقام تحدد معرفات أعضاء الجمعية.

لبرامج Python، لنقل `census.py` (يتم تشغيله عادةً كـ `pypy3 census.py`) قم بتشغيل أداة الاختبار كما يلي:

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

لبرامج C++، قم أولاً بتجميع حلك:

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

ثم قم بتشغيل أداة الاختبار:

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

لاحظ أنه في هذه المسألة يتم استخدام المخرجات القياسية (standard output) للتواصل مع المُقيّم، لذا لا ينبغي استخدامها لتصحيح الأخطاء (debugging). بدلاً من ذلك، يمكنك استخدام مخرجات الخطأ القياسية (stderr). في C++ يمكنك استخدام `cerr << msg << endl;`. في Python يمكنك استخدام `print(msg, file=sys.stderr)`.

ستقرأ أداة الاختبار رسائل الخطأ هذه وتعرضها جنباً إلى جنب مع الاستعلامات التي أجرتها جميع نسخ برنامجك. لاحظ أنه لأسباب فنية قد تظهر خارج التزامن قليلاً مع بعضها البعض.