الموازين

لدى أمينة ستة عملات معدنية، مرقمة من 1 حتى 6، وهي تعلم أن كل العملات مختلفة في الوزن، وتريد ترتيبهم بحسب وزنهم، ولهذا الغرض قامت بتطوير نوع جديد من الموازين.
الميزان العادي له كفتان، ولاستخدام هذا الميزان تقوم بوضع عملة في كل من الكفتين وسيقوم الميزان بتحديد أي عملة هي الأثقل.
ميزان أمينة الجديد معقد أكثر من ذلك، فهو يحوي أربع كفات، تسمى A،B،C،D. كما أن للميزان أربع إعدادات مختلفة في كل منها يجيب على سؤال مختلف عن العملات. لاستخدام الميزان يجب على أمينة أن تضع عملة واحدة تماماً في كل من الكفات A و B و C، بالإضافة إلى ذلك في الإعداد الرابع فقط يجب عليها أيضاً وضع عملة واحدة تماماً في الكفة D$.

ستطلب الإعدادات الأربعة من الميزان أن يجيب على الأسئلة الأربعة التالية:

1. أي من العملات في الكفات A،B،C هي الأثقل؟
2. أي من العملات في الكفات A،B،C هي الأخف؟
3. أي من العملات في الكفات A،B،C هي الوسيط؟ (أي العملة التي ليست الأثقل وليست الأخف بين العملات).
4. من بين العملات في الكفات A،B،C، لنأخذ بالاعتبار فقط العملات التي هي أثقل من العملة الموجودة في الكفة D. إذا وجد هكذا عملات سيتم إعطاء أي منها هي الأخف، وإلا إذا لم يكن هناك أي عملة أثقل من D, سيتم إعطاء أي من العملات في الكفات A،B،C هي الأخف.

المهمة

اكتب برنامجاً يقوم بترتيب عملات أمينة الستة بحسب وزنهم. يمكن للبرنامج أن يستعلم ميزان أمينة لمقارنة أوزان العملات.
سيتم اختبار برنامجك على عدة حالات اختبار في كل تشغيل يمثل كل منها مجموعة جديدة من العملات الستة.
يجب على برنامجك أن يحقق التابع `init` و `orderCoins`. خلال كل تشغيل لبرنامجك سيقوم المصحح أولاً باستدعاء التابع `init` تماماً مرة واحدة، هذا ما سيعطيك عدد حالات الاختبار ويسمح لك بتهيئة المتحولات. ثم سيقوم المصحح باستدعاء التابع `orderCoins()` مرة واحدة من أجل كل حالة اختبار.

- `init(T)`
  - `T`: عدد حالات الاختبار التي يجب على برنامجك أن يحلها خلال هذا التشغيل، `T` هو عدد صحيح ضمن المجال $1,\ldots,18$.
  - هذا التابع لا يعيد أي قيمة.
- `orderCoins()`
  - يتم استدعاء هذا التابع مرة واحدة تماماً من أجل كل حالة اختبار.
  - يجب على التابع أن يحدد الترتيب الصحيح لعملات أمينة عن طريق استدعاء توابع المصحح `getLightest()`, `getHeaviest()`, `getMedian()`,`getNextLightest()`. أي منها أو كلها.
  - حالما يعلم التابع الترتيب الصحيح يجب عليه الإعلام عن ذلك عن طريق استدعاء التابع `answer()`.

*بعد استدعاء `answer()`, يجب على التابع `orderCoins()` أن ينتهي وهو لا يعيد أي قيمة.

يمكنك استخدام توابع المصحح التالية في برنامجك:

- answer(W) — يجب على برنامجك استخدام هذا التابع للإعلام بأنه قد أوجد الجواب.
  - W: مصفوفة من ستة عناصر تحوي الترتيب الصحيح للعملات، W[0] حتى W[5] يجب أن تكون أرقام العملات (أي الأرقام من 1 وحتى 6) مرتبة من الأخف وحتى الأثقل.

  *يجب على برنامجك استدعاء هذا التابع مرة واحدة فقط ضمن التابع orderCoins() خلال حالة الاختبار الواحدة.
  - لا يعيد هذا التابع أي قيمة.
- getHeaviest(A, B, C), getLightest(A, B, C), getMedian(A, B, C) — هذه التوابع توافق الإعدادات 1 و 2و 3 على التتالي من أجل ميزان أمينة.
  - A, B, C: العملات الموضوعة ضمن الكفات A،B و C، على التوالي ، A،B و C يجب أن تكون ثلاثة أعداد صحيحة مختلفة وكل منها بين 1 و 6.

  *يعيد كل تابع واحداً من الأعداد A، B،C: رقم العملة المناسبة مثلاً getHeaviest(A, B, C) يعيد رقم العملة الأثقل بين العملات الثلاثة.
- getNextLightest(A, B, C, D) — يتوافق هذا التابع مع الإعداد الرابع من ميزان أمينة:
  - A, B, C, D: العملات التي تم وضعها في كل من A, B, C, D, على التوالي . A, B, C, D يجب أن تكون أربع أعداد مختلفة كل منها ضمن المجال من 1 وحتى 6.

  *يعيد هذا التابع أحد الأعداد A, B, و C: وهو رقم العملة التي يتم تحديدها بالميزان كما هو مشروح أعلاه في الإعداد 4 أي: يعيد أخف عملة بين العملات الموجودة على الكفات A, B, و C والتي هي أثقل من العملة الموضوعة في الكفة D؛ إذا لم يكن أي منها أثقل من العملة في الكفة D، العملة التي يتم إعادة رقمها هي ببساطة العملة الأخف بين العملات A, B, and C.

التقييم

لا يوجد مهمات جزئية في هذه المسألة، بدلاً من ذلك سيكون تقييمك معتمداً على عدد المرات التي قمت بها بالوزن (أي عدد مرات استدعاء التوابع getLightest(), getHeaviest(), getMedian() and/or getNextLightest()) التي قام بها برنامجك.
سيتم تشغيل برنامجك عدة مرات وفي كل مرة سيكون هناك أكثر من حالة استخدام، ليكن r عدد مرات تشغيل البرنامج ، هذا الرقم متعلق ببيانات الاختبار، إذا لم يقم برنامجك بترتيب العملات بشكل صحيح في واحدة من حالات الاختبار في تشغيل معين ستحصل على 0 نقطة، وإلا سيتم احتساب نقاط كل تشغيل كما يلي:

ليكن Q هو أصغر عدد بحيث أنه من الممكن ترتيب أي تسلسل من ست عملات باستخدام Q عملية وزن على ميزان أمينة.
لتكون المسالة أكثر تحدياً لن نفصح عن الرقم Q هنا.

لنفترض أن أكبر رقم من عمليات الوزن في كل حالات الاختبار في كل التشغيلات هو $Q + y$ من أجل عدد صحيح معين y.
وهكذا من أجل تشغيل واحد لبرنامجك ليكن أكبر عدد من عمليات الوزن بين كل T حالة اختبار في هذا التشغيل هو $Q + x$ من أجل عدد صحيح غير سالب x. (إذا قمت باستخدام عدد أقل من Q عملية وزن من أجل كل حالات الاختبار عندئذ يكون $x = 0$)عندئذ ستكون نتيجتك كالتالي: $\frac{100}{r((x+y)/5+1)}$, مقربة إلى أقل عددين بعد الفاصلة العشرية.
بشكل خاص إذا قام برنامجك على الأكثر ب Q عملية وزن في كل حالات الاختبار في كل عمليات التشغيل ستحصل على علامة 100.

مثال:

ليكن ترتيب العملات هو **3 4 6 2 1 5** من الأخف إلى الأثقل (ومن اليسار لليمين).

الاستدعاء	يعيد	الشرح
getMedian(4, 5, 6)	6	Coin **6** is the median among coins **4**, **5**, and **6**.
getHeaviest(3, 1, 2)	1	Coin **1** is the heaviest among coins **1**, **2**, and **3**.
getNextLightest(2, 3, 4, 5)	3	Coins **2**, **3**, and **4** are all lighter than coin **5**, so the lightest among them (**3**) is returned.
getNextLightest(1, 6, 3, 4)	6	Coins **1** and **6** are both heavier than coin **4**. Among coins **1** and **6**, coin **6** is the lightest one.
getHeaviest(3, 5, 6)	5	Coin **5** is the heaviest among coins **3**, **5** and **6**.
getMedian(1, 5, 6)	1	Coin **1** is the median among coins **1**, **5** and **6**.
getMedian(2, 4, 6)	6	Coin **6** is the median among coins **2**, **4** and **6**.
answer([3, 4, 6, 2, 1, 5])		The program found the right answer for this test case.

نموذج المصحح

يقوم نموذج المصحح بقراءة الدخل وفق الترتيب التالي:

- line 1: T — the number of test cases
- each of the lines from 2 to $T + 1$: a sequence of 6 distinct numbers from 1 to 6: the order of the coins from the lightest to the heaviest.

For instance, an input that consists of two test cases where the coins are ordered **1 2 3 4 5 6** and **3 4 6 2 1 5** looks as follows:

```
2
1 2 3 4 5 6
3 4 6 2 1 5
```

The sample grader prints the array that was passed as a parameter to the `answer()` function.