Waage

Amina hat sechs Münzen, die von 1 bis 6 nummeriert sind. Sie weiß, dass alle Münzen unterschiedliches Gewicht haben. Amina möchte sie nach ihrem Gewicht ordnen. Für diesen Zweck hat sie eine neuartige Balkenwaage entwickelt.

Eine herkömmliche Balkenwaage hat zwei Waagschalen. Um eine derartige Waage zu benutzen, legt man in jede Waagschale eine Münze. Die Waage zeigt danach an, welche Münze schwerer ist.

Aminas neue Waage ist wesentlich komplexer. Sie hat vier Waagschalen, die mit A, B, C und D bezeichnet werden. Die Waage hat vier verschiedene Einstellmöglichkeiten. Jede Einstellung liefert je nach Fragestellung bezüglich der Münzen eine entsprechende Antwort. Um die Waage zu benutzen, muss Amina genau eine Münze in jede Waagschale A, B und C legen. In der vierten Einstellung muss zusätzlich genau eine Münze in die Waagschale D gelegt werden.

Die vier Einstellungen bewirken, dass die Waage die folgenden vier Fragen beantwortet:

1. Welche der Münzen in den Waagschalen A, B und C ist die schwerste?
2. Welche der Münzen in den Waagschalen A, B und C ist die leichteste?
3. Welche der Münzen in den Waagschalen A, B und C ist der Median? (Das ist die Münze, welche von den dreien weder die leichteste noch die schwerste ist.)
4. Von den Münzen in den Waagschalen A, B und C werden nur jene Münzen berücksichtigt, die schwerer sind als die Münze in der Waagschale D. Wenn es solche Münzen gibt, wird die leichteste in der Antwort ausgegeben. Anderenfalls, wenn es keine solche Münzen gibt, wird als Antwort die leichteste von den Waagschalen A, B und C in der Antwort ausgegeben.

Aufgabe

Schreibe ein Programm, das Aminas sechs Münzen gemäß dem Gewicht ordnet (die leichteste zuerst). Das Programm kann Aminas Waage zum Vergleichen der Gewichte der Münzen benutzen. Deinem Programm werden verschiedene Testfälle zum Lösen übergeben, wobei jeder Testfall ein neues Satz von Münzen darstellt.

Dein Programm sollte die Funktionen `init` und `orderCoins` implementieren. Während jedes Programmdurchlaufs ruft der Grader zuerst genau einmal die Funktion `init` auf. Das liefert die Anzahl der Testfälle und erlaubt dir, alle Variablen zu initialisieren. Der Grader wird dann einmal pro Testfall die Funktion `orderCoins()` aufrufen.

- `init(T)`
  - `T`: Die Anzahl der Testfälle, die dein Programm während dieses Durchlaufs zu lösen hat. `T` ist ein Integer im Bereich von $1, \ldots, 18$.
  - Diese Funktion hat keinen Rückgabewert.

- `orderCoins()`
  - Diese Funktion wird genau einmal pro Testfall aufgerufen.
  - Diese Funktion soll die richtige Reihenfolge von Aminas Münzen ermitteln, indem sie die Grader-Funktionen `getHeaviest()`, `getLightest()`, `getMedian()` und/oder `getNextLightest()` aufruft.
  - Sobald die Funktion die richtige Reihenfolge kennt, soll sie diese durch Aufruf der Grader-Funktion `answer()` mitteilen.
  - Nach dem Aufruf von `answer()` sollte die Funktion `orderCoins()` beendet werden. Sie hat keinen Rückgabewert.

Du kannst in deinem Programm folgende Graderfunktionen benutzen:

- `answer(W)` — dein Programm sollte diese Funktion benutzen, um die gefundene Antwort mitzuteilen.
  - `W`: Ist ein Array der Länge 6, welches die Münzen in der richtigen Reihenfolge enthält. `W[0]` bis `W[5]` sollten die Nummern der Münzen sein (das heißt, Zahlen von 1 bis 6), geordnet von der leichtesten zur schwersten Münze.
  - Dein Programm sollte diese Funktion innerhalb der Funktion `orderCoins()` einmal pro Testfall aufrufen.
  - Diese Funktion hat keinen Rückgabewert.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — diese Funktionen entsprechen den Einstellungen 1, 2 und 3 von Aminas Waage.
  - `A`, `B`, `C`: Die Münzen, die in die Waagschalen A, B und C gelegt werden. `A`, `B`, and `C` sollen drei verschiedene Integer sein, die zwischen 1 and 6 (inklusive) liegen.
  - Jede Funktion gibt eine der drei Zahlen `A`, `B` und `C` zurück: Die Nummer der entsprechenden Münze. Beispielsweise gibt `getHeaviest(A, B, C)` die Nummer der schwersten der drei untersuchten Münzen zurück.
- `getNextLightest(A, B, C, D)` — diese Funktion entspricht der Einstellung 4 von Aminas Waage.
  - `A`, `B`, `C`, `D`: Die Münzen, die in die Waagschalen A, B , C und D gelegt werden. `A`, `B`, `C` und `D` sollen vier verschiedene Integer sein, die zwischen 1 and 6 (inklusive) liegen.
  - Die Funktion gibt eine der drei Zahlen `A`, `B` und `C` zurück: Die Zahl, die die Waage in der oben beschriebenen Einstellung 4 liefert. Das bedeutet, die zurückgegebene Münze ist die leichteste unter den Münzen in den Waagschalen A, B und C, die schwerer sind als die Münze in der Waagschale D. Oder: wenn keine schwerer ist als die Münze in der Waagschale D, dann ist die zurückgegebene Münze einfach die leichteste von den drei Münzen in den Waagschalen A, B und C.

Bewertung

Diese Aufgabe hat keine Teilaufgaben. Deine Punktzahl hängt davon ab, wie oft dein Programm "wiegen" lässt — also von der Gesamtanzahl der Aufrufe von `getLightest()`, `getHeaviest()`,

`getMedian()` und `getNextLightest()`. Jeder dieser Aufrufe wird auch "Wägung" genannt.

Es gibt r Testläufe mit jeweils mehreren Testfällen. Falls dein Programm in irgendeinem Testfall die Münzen nicht richtig sortiert, bekommst du 0 Punkte. Ansonsten wird jeder Lauf einzeln bewertet, und zwar wie folgt:

Sei Q die kleinste Zahl mit der Eigenschaft, dass man beliebige sechs Münzen mit Q Wägungen sortieren kann. Q wird hier natürlich nicht verraten.

Die größte Anzahl von Wägungen, die dein Programm in einem Testfall benötigt hat (über alle Testfälle und alle Testläufe), sei $Q + y$ (schlechtestes Ergebnis insgesamt). Für einen bestimmten Testlauf sei wiederum $Q + x$ die größte Anzahl von Wägungen, die dein Programm in einem Testfall dieses Testlaufs benötigt hat (schlechtestes Ergebnis in diesem Testlauf; solltest du in jedem Testfall weniger als Q Wägungen benötigen, ist $x = 0$). Dann ist die Punktzahl für diesen Lauf: $\frac{100}{r((x+y)/5+1)}$ (abgerundet auf zwei Stellen nach dem Komma).

Falls dein Programm in jedem Testfall jedes Testlaufs höchstens Q Wägungen benötigt, erhältst du folglich 100 Punkte.

Beispiel

Die Sortierung der Münzen 1 bis 6 nach Gewicht (leichteste zuerst) ergibt die Reihenfolge $3\ 4\ 6\ 2\ 1\ 5$.

Aufruf	Rückgabe	Erklärung
getMedian(4, 5, 6)	6	Münze 6 ist der Median der Münzen 4, 5 und 6.
getHeaviest(3, 1, 2)	1	Münze 1 ist die schwerste der Münzen 1, 2 und 3.
getNextLightest(2, 3, 4, 5)	3	Die Münzen 2, 3 und 4 sind alle leichter als Münze 5. Die leichteste dieser drei Münzen ist 3.
getNextLightest(1, 6, 3, 4)	6	Nur die Münzen 1 und 6 sind schwerer als Münze 4. Von diesen beiden wiederum ist Münze 6 die leichteste.
getHeaviest(3, 5, 6)	5	Münze 5 ist die schwerste der Münzen 3, 5 und 6.
getMedian(1, 5, 6)	1	Münze 1 ist der Median der Münzen 1, 5 und 6.
getMedian(2, 4, 6)	6	Münze 6 ist der Median der Münzen 2, 4 und 6.
answer([3, 4, 6, 2, 1, 5])		Das Programm hat (für diesen Testfall) die richtige Antwort berechnet.

Beispielgrader

Der Beispielgrader liest die Eingabe im folgenden Format ein:

- Zeile 1: T —- Anzahl der Testfälle
- Zeilen 2 bis $T + 1$: eine Permutation der Zahlen 1 bis 6 —- die Reihenfolge der nach Gewicht sortierten Münzen (leichteste zuerst).

Für einen Programmlauf mit zwei Testfällen, in denen die sortierte Reihenfolge der Münzen $1\ 2\ 3\ 4\ 5\ 6$ bzw. $3\ 4\ 6\ 2\ 1\ 5$ ist, sieht die Eingabe so aus:

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

Der Beispielgrader gibt den Array aus, den dein Programm der Funktion `answer()` als Parameter übergeben hat.