Krzysztof Diks, Lech Duraj
Tłumaczenie

Dostępna pamięć: 1500 MB, limit czasu: 1 s. *IOI 2015, dzień pierwszy, 28.07.2015*

Waga szalkowa

Amina ma sześć monet, ponumerowanych od 1 do 6. Wiadomo, że wszystkie monety mają różne masy (ciężary). Amina chce uporządkować monety rosnąco względem ich mas. W tym celu zbudowała wagę szalkową nowego typu.

Tradycyjna waga szalkowa ma dwie szalki. Korzystanie z takiej wagi polega na umieszczeniu po jednej monecie na każdej z szalek i wskazaniu cięższej z nich.

Nowa waga Aminy jest bardziej skomplikowana. Waga ma cztery szalki, oznaczone A, B, C, D, i pracuje w czterech trybach, w każdym odpowiadając na inne pytanie dotyczące monet położonych na szalkach. Użycie wagi polega na umieszczeniu po jednej monecie na każdej z szalek A, B i C. Dodatkowo, w trybie czwartym, należy umieścić także monetę na szalce D.

W każdym z trybów 1–4 dostajemy odpowiedź na pytanie dotyczące monet umieszczonych na szalkach:

1. *Która z monet na szalkach A, B i C jest najcięższa?*
2. *Która z monet na szalkach A, B i C jest najlżejsza?*
3. *Która z monet na szalkach A, B i C jest medianą? (Medianą jest ta moneta wśród trzech, która nie jest ani najcięższa, ani najlżejsza).*
4. *Wśród monet na szalkach A, B i C rozważamy tylko te, które są cięższe od tej z szalki D. Jeśli są takie monety, to która z nich jest najlżejsza? W przeciwnym przypadku (gdy nie ma takich monet), która z monet na szalkach A, B i C jest najlżejsza?*

Zadanie

Napisz program, który uporządkuje sześć monet Aminy rosnąco względem ich mas. Program może korzystać z wagi Aminy do porównywania mas monet. Twój program będzie musiał rozwiązać wiele przypadków testowych, każdy odpowiadający nowemu zestawowi sześciu monet.

W Twoim programie powinny zostać zaimplementowane funkcje `init` *oraz* `orderCoins`*. W każdym wykonaniu Twojego programu, program sprawdzający wywoła najpierw, dokładnie raz, funkcję* `init`*. Jako jej parametr podana zostanie liczba przypadków testowych, a dodatkowo będziesz mógł zainicjować dowolne zmienne. Następnie program sprawdzający będzie wywoływał funkcję* `orderCoins` *raz dla każdego przypadku testowego.*

- `init(T)`
  - `T`*: Liczba przypadków testowych, które program będzie musiał rozwiązać podczas tego przebiegu.* `T` *jest liczbą całkowitą z zakresu* $1, \ldots, 18$*.*
  - *Funkcja nie zwraca żadnej wartości.*
- `orderCoins()` *– funkcja wywoływana dokładnie raz dla każdego przypadku testowego.*
  - *Ta funkcja powinna wyznaczyć poprawne uporządkowanie monet Aminy, wywołując funkcje programu sprawdzającego* `getLightest()`*,* `getHeaviest()`*,* `getMedian()` *i/lub* `getNextLightest()`*.*

- *Gdy funkcja wyznaczy właściwe uporządkowanie, powinna je podać, wywołując funkcję programu sprawdzającego* `answer()`.
- *Po wywołaniu* `answer()`, *funkcja* `orderCoins()` *powinna zakończyć swoje działanie. Funkcja nie zwraca żadnej wartości.*

W swoim programie możesz korzystać z następujących funkcji programu sprawdzającego:

- `answer(W)` – *Twój program powinien użyć tej funkcji do podania obliczonej odpowiedzi.*
  - `W`*: Tablica o długości 6, zawierająca poprawne uporządkowanie monet. Elementy tablicy od* `W[0]` *do* `W[5]` *powinny zawierać numery monet (czyli numery od 1 do 6) w kolejności odpowiadającej monetom od najlżejszej do najcięższej.*
  - *Twój program powinien wywołać tę funkcję z* `orderCoins()`*, raz dla każdego przypadku testowego.*
  - *Ta funkcja nie zwraca żadnej wartości.*
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` – *te funkcje odpowiadają użyciom wagi Aminy odpowiednio w trybach 1, 2 i 3.*
  - `A`, `B`, `C`*: Monety, które umieszcza się odpowiednio na szalkach A, B i C.* `A`, `B` *i* `C` *powinny być różnymi liczbami całkowitymi z zakresu od 1 do 6 włącznie.*
  - *Każda z tych funkcji zwraca jedną z liczb* `A`, `B` *i* `C`*: numer właściwej monety. Dla przykładu,* `getHeaviest(A, B, C)` *zwraca numer najcięższej z tych trzech monet.*
- `getNextLightest(A, B, C, D)` – *ta funkcja odpowiada trybowi 4 wagi Aminy.*
  - `A`, `B`, `C`, `D`*: Monety, które umieszcza się odpowiednio na szalkach A, B, C i D.* `A`, `B`, `C` *i* `D` *powinny być różnymi liczbami całkowitymi z zakresu od 1 do 6 włącznie.*
  - *Funkcja zwraca jedną z liczb* `A`, `B` *i* `C`*: numer monety wskazanej przez wagę pracującą w trybie 4. Innymi słowy, zwracana moneta jest najlżejszą spośród tych monet na szalkach A, B i C, które są cięższe niż moneta na szalce D; lub, jeśli wszystkie te monety są lżejsze od monety na szalce D, zwracana jest najlżejsza z monet na szalkach A, B i C.*

Podzadania

To zadanie nie posiada żadnych podzadań. Zamiast tego Twoja punktacja zależy od liczby ważeń, które wykona program (łączna liczba wywołań funkcji `getLightest()`, `getHeaviest()`, `getMedian()` *i/lub* `getNextLightest()`*).*

Twój program będzie wykonywany wiele razy, za każdym razem dla wielu przypadków testowych. Niech r będzie liczbą wykonań Twojego programu (liczbą testów w zadaniu). Jeśli Twój program nie znajdzie poprawnej kolejności monet dla któregokolwiek przypadku testowego w jakimkolwiek wykonaniu programu, otrzymasz 0 punktów. W przeciwnym przypadku poszczególne wykonania programu są punktowane następująco.

Niech Q będzie najmniejszą liczbą ważeń, która umożliwia posortowanie dowolnego ciągu sześciu monet na wadze Aminy. Żeby uczynić zadanie bardziej interesującym, nie podajemy tutaj wartości Q.

Załóżmy, że największa liczba ważeń wykonanych przez Twój program dla wszystkich podanych przypadków testowych wynosi $Q + y$, dla pewnej liczby całkowitej y. Rozważmy teraz pojedyncze wykonanie programu. Niech największa liczba ważeń dla wszystkich T przypadków testowych w tym wykonaniu wynosi $Q + x$, dla pewnej nieujemnej liczby całkowitej x. (Jeśli użyjesz mniej niż Q ważeń dla każdego przypadku testowego, wówczas $x = 0$). Wówczas liczba punktów zdobyta za to wykonanie wyniesie $\frac{100}{r((x+y)/5+1)}$, zaokrąglone **w dół** *do dwóch cyfr po przecinku.*

W szczególności, jeśli Twój program wykona co najwyżej Q ważeń dla każdego przypadku testowego w każdym wykonaniu programu, otrzymasz 100 punktów.

Przykład

Załóżmy, że kolejność monet od najlżejszej do najcięższej jest następująca: 3 4 6 2 1 5.

wywołania funkcji	**wyniki**	**objaśnienie**
`getMedian(4, 5, 6)`	*6*	*Moneta 6 jest medianą wśród monet 4, 5 i 6.*
`getHeaviest(3, 1, 2)`	*1*	*Moneta 1 jest najcięższą wśród monet 1, 2 i 3.*
`getNextLightest(2, 3, 4, 5)`	*3*	*Monety 2, 3 i 4 są wszystkie lżejsze od monety 5, zatem zostanie zwrócona najlżejsza spośród nich (3).*
`getNextLightest(1, 6, 3, 4)`	*6*	*Monety 1 i 6 są obie cięższe niż moneta 4. Spośród monet 1 i 6, moneta 6 jest najlżejsza.*
`getHeaviest(3, 5, 6)`	*5*	*Moneta 5 jest najcięższą wśród monet 3, 5 i 6.*
`getMedian(1, 5, 6)`	*1*	*Moneta 1 jest medianą wśród monet 1, 5 i 6.*
`getMedian(2, 4, 6)`	*6*	*Moneta 6 jest medianą wśród monet 2, 4 i 6.*
`answer([3, 4, 6, 2, 1, 5])`		*Program znalazł właściwą odpowiedź dla tego przypadku testowego.*

Przykładowy program sprawdzający

Przykładowy program sprawdzający wczytuje dane w następującym formacie:

- *wiersz 1:* `T` *– liczba przypadków testowych*
- *każdy z wierszy od 2 do* `T` *+ 1: ciąg 6 różnych liczb od 1 do 6: kolejność monet od najlżejszej do najcięższej.*

Dla przykładu, dla dwóch przypadków testowych, w których monety są dane odpowiednio w kolejności 1 2 3 4 5 6 i 3 4 6 2 1 5, dane mają postać:

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

Przykładowy program sprawdzający wypisuje zawartość tablicy, która została przekazana jako parametr do funkcji `answer()`.