Weegschaal

Amina heeft zes munten, genummerd van 1 tot 6. Ze weet dat alle munten een verschillend gewicht hebben. Ze wil de munten ordenen volgens hun gewicht. Daar heeft ze een nieuw soort weegschaal voor ontworpen.

Een traditionele weegschaal heeft twee pannen. Om zo'n schaal te gebruiken, plaats je een munt in iedere pan en de schaal bepaalt welke munt zwaarder is.

Amina's schaal is complexer. Ze heeft vier pannen, gelabeld A, B, C, en D. De schaal heeft vier verschillende configuraties, die elk een andere vraag beantwoorden over de munten. Om de schaal te gebruiken moet Amina exact één munt in elk van de pannen A, B, en C plaatsen. In de vierde configuratie moet ze daarbovenop ook nog exact één munt in pan D plaatsen.

De vier configuraties laten de schaal de volgende vier vragen beantwoorden:

1. Welke van de munten in pannen A, B, en C is de zwaarste?
2. Welke van de munten in pannen A, B, en C is de lichtste?
3. Welke van de munten in pannen A, B, en C is de mediaan? (Dit is de munt die noch de zwaarste noch de lichtste is van de drie.)
4. Gegeven de munten in pannen A, B, en C, beschouw enkel de munten die zwaarder zijn dan de munt in pan D. Als er zulke munten zijn, welke van deze munten is dan de lichtste? Anders, als er geen zulke munten zijn, welke van de munten in pannen A, B, en C is dan de lichtste?

Taak

Schrijf een programma dat Amina's zes munten zal sorteren volgens hun gewicht. Het programma kan Amina's schaal bevragen om de gewichten van de munten te vergelijken. Je programma zal verschillende testcases moeten oplossen, die elk overeenkomen met een nieuwe set van zes munten.

Je programma moet de functies `init` and `orderCoins` implementeren. Tijdens elke uitvoering van je programma, zal de grader eerst exact één keer `init` aanroepen. Dit geeft je het aantal testcases en laat je toe om variabelen te initialiseren. Daarna roept de grader éénmaal per testcase `orderCoins()` aan.

- `init(T)`
  - `T`: Het aantal testcases dat je programma zal moeten oplossen tijdens deze uitvoering. `T` is een integer in het interval $1, \ldots, 18$ (inclusief).
  - Deze functie geeft geen waarde terug.
- `orderCoins()`
  - Deze functie wordt exact éénmaal per testcase aangeroepen.

- De functie moet de correcte volgorde van Amina's munten bepalen door de volgende functies van de grader aan te roepen: `getHeaviest()`, `getLightest()`, `getMedian()`, en/of `getNextLightest()`.
- Eens de functie de correcte volgorde weet, moet dit worden gemeld door de graderfunctie `answer()` aan te roepen.
- Na het aanroepen van `answer()`, moet de functie `orderCoins()` eindigen. Ze geeft geen waarde terug.

Je mag de volgende graderfuncties in je programma gebruiken:

- `answer(W)` — je programma moet deze functie gebruiken om het antwoord te rapporteren dat werd gevonden.
  - `W`: Een array van lengte 6 die de correcte volgorde van munten bevat. `W[0]` tot en met `W[5]` moeten muntnummers bevatten (dus: getallen van **1** tot **6**) in volgorde van de lichtste tot de zwaarste munt.
  - Je programma mag deze functie enkel aanroepen vanuit `orderCoins()`, en dat éénmaal per testcase.
  - Deze functie geeft geen waarde terug.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — deze komen respectievelijk overeen met configuraties 1, 2 en 3 op Amina's weegschaal.
  - `A`, `B`, `C`: De munten die respectievelijk in pannen A, B, and C worden geplaatst. `A`, `B`, en `C` moeten drie verschillende integers zijn, elk tussen **1** en **6** (inclusief).
  - Elke functie geeft één van de getallen `A`, `B` of `C` terug: het getal dat overeenkomt met de juiste munt. Bijvoorbeeld, `getHeaviest(A, B, C)` geeft het nummer terug van de zwaarste van de drie gegeven munten.
- `getNextLightest(A, B, C, D)` — dit komt overeen met configuratie 4 op Amina's weegschaal
  - `A`, `B`, `C`, `D`: De munten in respectievelijk in pannen A, B, C, en D worden geplaatst. `A`, `B`, `C`, en `D` moeten vier verschillende integers zijn, elk tussen **1** en **6** (inclusief).
  - De functie geeft één van de nummers `A`, `B`, of `C` terug: het nummer van de munt geselecteerd door de schaal zoals hierboven beschreven voor configuratie 4. Dus: de teruggegeven munt is die lichtste van de munten op pannen A, B, en C die zwaarder zijn dan de munt in pan D; of als geen ervan zwaarder is dan de munt in pan D, dan is de teruggegeven munt gewoon de lichtste van de drie munten op pannen A, B, en C.

Score

Er zijn geen subtaken voor dit probleem. In plaats daarvan wordt je score bepaald door hoeveel wegingen (het totaal aantal aanroepen van graderfuncties `getLightest()`, `getHeaviest()`, `getMedian()` en/of `getNextLightest()`) je programma maakt.

Je programma wordt verschillende keren uitgevoerd met verschillende testcases per uitvoering. Laat r het aantal uitvoeringen zijn van je programma. Dit nummer is bepaald door de testdata. Als je

programma de munten niet correct sorteert in eender welke testcase van eender welke uitvoering, krijg je 0 punten. In het andere geval worden de uitvoeringen individueel beoordeeld als volgt.

Laat Q het kleinst mogelijke getal zijn zodat het mogelijk is om eender welke reeks van zes munten te sorteren met Amina's schaal in Q wegingen. Om deze taak wat uitdagender te maken, zeggen we je niet wat de waarde van Q is.

Stel dat het grootste aantal wegingen onder alle testcases van alle uitvoeringen gelijk is aan $Q + y$ voor een of andere integer y. Beschouw dan één enkele uitvoering van je programma. Laat het grootste aantal wegingen onder alle T testcases in deze uitvoering gelijk zijn aan $Q + x$ voor een niet-negatieve integer x. (Als je minder dan Q wegingen gebruikt voor elke testcase, dan $x = 0$.). Dan is de score voor deze uitvoering gelijk aan $\frac{100}{r((x+y)/5+1)}$, afgerond *naar beneden* tot twee getallen na de komma.

Meer specifiek, als jouw programma ten hoogste Q wegingen maakt in elke testcase van elke uitvoering, dan krijg je 100 punten.

Voorbeeld

Stel dat de munten geordend zijn als $3\ 4\ 6\ 2\ 1\ 5$ van de lichtste tot de zwaarste.

Functie-aanroep	Returnwaarde	Uitleg
getMedian(4, 5, 6)	6	Munt **6** is de mediaan van de munten **4**, **5**, en **6**.
getHeaviest(3, 1, 2)	1	Munt **1** is de zwaarste van de munten **1**, **2**, en **3**.
getNextLightest(2, 3, 4, 5)	3	Munten **2**, **3**, en **4** zijn allen lichter dan munt **5**, dus de lichtste van de drie (**3**) wordt teruggegeven.
getNextLightest(1, 6, 3, 4)	6	Munten **1** en **6** zijn beide zwaarder dan munt **4**. Van de munten **1** en **6**, is munt **6** de lichtste.
getHeaviest(3, 5, 6)	5	Munt **5** is de zwaarste van de munten **3**, **5** en **6**.
getMedian(1, 5, 6)	1	Munt **1** is de mediaan van de munten **1**, **5** en **6**.
getMedian(2, 4, 6)	6	Munt **6** is de mediaan van de munten **2**, **4** en **6**.
answer([3, 4, 6, 2, 1, 5])		Het programma heeft het juiste antwoord gevonden voor deze testcase.

Voorbeeldgrader

De voorbeeldgrader leest input in het volgende formaat:

- lijn 1: T —- het aantal testcases
- elk van de lijnen 2 tot en met $T + 1$: een reeks van 6 verschillende getallen van 1 tot en met 6: de volgorde van de munten van de lichtste tot de zwaarste.

Bijvoorbeeld, een input die bestaat uit twee testcases waar de munten geordend zijn als $1\ 2\ 3\ 4\ 5\ 6$ en $3\ 4\ 6\ 2\ 1\ 5$ ziet eruit als volgt:

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

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

De voorbeeldgrader print de array die als parameter werd gegeven aan de functie `answer()`.