



## Scales

Amina heeft zes munten, genummerd van **1** tot en met **6**. Alle munten hebben een uniek gewicht. Ze wil de munten sorteren op volgorde van gewicht. Hiervoor heeft ze een nieuw soort weegschaal bedacht.

Een traditionele weegschaal heeft twee bakjes. Om deze te gebruiken leg je in beide bakjes een munt. De weegschaal bepaalt welke munt het zwaarste is.

De weegschaal van Amina is complexer. Deze heeft vier bakjes, met labels **A**, **B**, **C** en **D**. De weegschaal heeft ook vier verschillende instellingen. Elke instelling zorgt ervoor dat een andere vraag over de munten wordt beantwoord. Om de weegschaal te gebruiken moet Amina precies één munt plaatsen in elk van de bakjes **A**, **B** en **C**. Bij de vierde instelling moet ze ook nog precies één munt plaatsen in bakje **D**.

De vier instellingen bepalen welke van de volgende vragen beantwoord wordt:

1. Welke munt in de bakjes **A**, **B** en **C** is het zwaarst?
2. Welke munt in de bakjes **A**, **B** en **C** is het lichtst?
3. Welke munt in de bakjes **A**, **B** en **C** is de mediaan? (Dit is de munt die niet het zwaarste en ook niet het lichtste is van de drie.)
4. Kijk in de bakjes **A**, **B** en **C** alleen naar de munten die zwaarder zijn dan de munt in bakje **D**. Als er zwaardere munten zijn, geef dan de lichtste van deze. Als er geen zwaardere munten zijn, geef dan de lichtste munt uit de bakjes **A**, **B** en **C**.

## Opdracht

Schrijf een programma dat de zes munten van Amina sorteert op gewicht. Je programma mag vragen stellen aan de weegschaal om de gewichten van de munten te vergelijken. Je programma krijgt verschillende testcases om op te lossen. Elke testcase komt overeen met een nieuwe set van zes munten.

Implementeer de functies `init` en `orderCoins`. Gedurende elke run van je programma roept de grader eerst `init` precies één keer aan. Zo weet je het aantal testcases en kun je je variabelen initialiseren. Daarna roept de grader de functie `orderCoins()` voor elke testcase één keer aan.

- `init(T)`
  - **T**: Het aantal testcases dat je programma moet oplossen. **T** heeft een waarde van **1** tot en met **18**.
  - Deze functie levert geen resultaat op.
- `orderCoins()`

- Voor elke testcase wordt deze functie precies eenmaal aangeroepen.
- Deze functie moet de correcte volgorde van de munten bepalen door de grader-functies `getLightest()`, `getHeaviest()`, `getMedian()`, en/of `getNextLightest()` aan te roepen.
- Zodra je functie de correcte volgorde heeft bepaald moet je dit doorgeven door de grader-functie `answer()` aan te roepen.
- Na de aanroep van `answer()`, moet de functie `orderCoins()` eindigen. Deze functie levert geen resultaat.

In je programma mag je de volgende grader-functies gebruiken:

- `answer(W)` — je programma moet deze functie gebruiken om door te geven wat het gevonden antwoord is.
  - `W`: Een array van lengte 6 met daarin de munten op de juiste volgorde. ‘`W[0]`’ tot en met ‘`W[5]`’ moeten de nummers van de munten bevatten (dus de getallen **1** tot en met **6**) op volgorde van lichtste tot zwaarste.
  - Je programma mag deze functie alleen aanroepen vanuit `orderCoins()` en één keer per testcase.
  - Deze functie levert geen resultaat op.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — deze functies horen bij de eerste, tweede en derde instelling van de weegschaal.
  - `A, B, C`: De munten die in de bakjes **A**, **B**, en **C** geplaatst worden. `A`, `B` en `C` zijn drie verschillende integers, elk van **1** tot en met **6**.
  - Elke functie levert als resultaat één van de aangeleverde nummers `A`, `B` en `C`: het nummer van de passende munt. Als voorbeeld retourneert `getHeaviest(A, B, C)` het nummer van de zwaarste van de drie gegeven munten.
- `getNextLightest(A, B, C, D)` — deze hoort bij vierde instelling van de weegschaal.
  - `A, B, C, D`: de munten die in de bakjes **A**, **B**, **C**, en **D** geplaatst zijn. `A`, `B`, `C`, en `D` moeten vier verschillende integers, elk van **1** tot en met **6** zijn.
  - De functie levert als resultaat één van de getallen `A`, `B` en `C`: het nummer van de munt die door de weegschaal is bepaald conform de hierboven beschreven vierde instelling. Dus: de munt is de lichtste van de munten in de bakjes **A**, **B** en **C** die zwaarder zijn dan de munt in bakje **D**; of, als geen van de munten zwaarder is dan de munt in bakje **D**, de lichtste van de drie munten in de bakjes **A**, **B** en **C**.

## Score

Bij dit probleem zijn geen subtasks. Je score wordt bepaald door het aantal wegingen (het totaal aantal aanroepen van de grader functies `getLightest()`, `getHeaviest()`, `getMedian()` en/of `getNextLightest()`) dat je programma doet.

Je programma wordt meerdere keren uitgevoerd met meerdere testcases in elke run. *r* is het aantal runs van je programma. Dit wordt bepaald door de testdata. Als je programma voor een testcase in

een van de runs niet het juiste antwoord doorgeeft krijg je 0 punten. Anders wordt je score als volgt bepaald:

$Q$  is het kleinste getal zodat het mogelijk is om elke rij van zes munten te sorteren met maximaal  $Q$  wegingen van de weegschaal. Je krijgt de waarde van  $Q$  niet te horen.

Stel dat het maximale aantal wegingen onder alle testcases van al jouw runs  $Q + y$  is. Kijk dan naar een enkele run van je programma. Stel dat het maximale aantal wegingen van de  $T$  testcases van deze run  $Q + x$  is met  $x$  niet negatief. (Als je voor elke testcase minder dan  $Q$  wegingen nodig hebt, dan geldt  $x = 0$ ). De score voor deze run is dan  $\frac{100}{r((x+y)/5+1)}$ , *omlaag* afgerond op twee decimalen.

In het bijzonder geldt dat als je programma voor elke testcase in elke run maximaal  $Q$  wegingen nodig heeft, je 100 punten krijgt.

## Voorbeeld

Stel dat de munten als volgt gerangschikt zijn van licht naar zwaar: **3 4 6 2 1 5**.

| Functie aanroep             | Resultaat | 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         | Munt 2, 3, en 4 zijn alle lichter dan munt 5, dus de lichtste onder hen (3) wordt geretourneerd. |
| getNextLightest(1, 6, 3, 4) | 6         | Munt 1 en 6 zijn beide zwaarder dan munt 4. Onder munten 1 en 6, is munt 6 de lichtste.          |
| getHeaviest(3, 5, 6)        | 5         | Munt 5 is de zwaarste onder munten 3, 5 en 6.                                                    |
| getMedian(1, 5, 6)          | 1         | Munt 1 is de mediaan onder munten 1, 5 en 6.                                                     |
| getMedian(2, 4, 6)          | 6         | Munt 6 is de mediaan onder munten 2, 4 en 6.                                                     |
| answer([3, 4, 6, 2, 1, 5])  |           | Het programma heeft het juiste antwoord voor deze testcase gevonden.                             |

## Voorbeeld grader

De voorbeeld grader leest invoer in het volgende formaat:

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

Een invoer van twee testcases waarbij de munten geordend zijn als **1 2 3 4 5 6** en **3 4 6 2 1 5** ziet er als volgt uit:

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

De voorbeeld grader drukt het array af dat als parameter is doorgegeven aan de functie `answer()`.