# Bilancia

Amina ha sei monete (numerate da $1$ a $6$) tutte di pesi diversi. Con l'intenzione di ordinarle per peso, ha sviluppato uno speciale tipo di bilancia a piatti. Una bilancia a piatti tradizionale ha due piatti: per usarla si mette una moneta in ciascun piatto e la bilancia determinerà quale delle due monete è la più pesante.

Invece, la nuova bilancia di Amina ha quattro piatti (chiamati $A$, $B$, $C$, e $D$) e quattro impostazioni differenti, ciascuna delle quali risponde a una domanda diversa riguardo alle monete. Per usare la bilancia, Amina deve posizionare esattamente una moneta in ciascuno dei piatti $A$, $B$, e $C$. Inoltre, se seleziona la quarta impostazione, deve anche posizionare esattamente una moneta nel piatto $D$.

Le quattro impostazioni rispondono alle seguenti domande:

1. Quale tra le monete nei piatti $A$, $B$, e $C$ è la più pesante?
2. Quale tra le monete nei piatti $A$, $B$, e $C$ è la più leggera?
3. Quale tra le monete nei piatti $A$, $B$, e $C$ è la mediana? (la moneta che non è né la più leggera né la più pesante)
4. Tra le monete nei piatti $A$, $B$, e $C$ consideriamo solo quelle che sono più pesanti della moneta nel piatto $D$. Se ve ne sono, quale di queste è la più leggera? Se invece non ve ne sono, qual è la più leggera in assoluto? (e che quindi si trova in uno dei piatti $A$, $B$, o $C$)

# Implementazione

Devi scrivere un programma che ordini le sei monete di Amina per peso crescente. Il programma può utilizzare la bilancia di Amina per confrontare il peso delle monete e dovrà risolvere diversi test case, ognuno corrispondente ad un nuovo insieme di sei monete.

Dovrai implementare le funzioni `init` e `orderCoins`. Durante ciascuna esecuzione del tuo programma, il grader chiamerà prima `init` esattamente una volta, fornendoti il numero di test case e consentendoti di inizializzare delle variabili. Il grader chiamerà quindi `orderCoins()` una volta per ogni test case.

- `init(T)`
  - `T`: Il numero di test case che il tuo programma dovrà risolvere durante questa esecuzione. `T` è un intero nell'intervallo $1, \ldots, 18$.
  - Questa funzione non ha valore di ritorno.
- `orderCoins()`
  - Questa funzione verrà chiamata esattamente una volta per test case.
  - La funzione deve determinare l'ordine corretto delle monete di Amina utilizzando le

funzioni fornite nel grader `getLightest()`, `getHeaviest()`, `getMedian()`, e/o `getNextLightest()`.

- Una volta che la funzione ha trovato l'ordine corretto, deve riportarlo chiamando la funzione del grader `answer()`.
- Dopo aver chiamato `answer()`, la funzione `orderCoins()` deve chiudersi. Non ha valore di ritorno.

Potrai utilizzare le seguenti funzioni definite nel grader:

- `answer(C)` — riporta la risposta che hai calcolato.
  - `C`: Un array di lunghezza $6$ contenente il corretto ordine delle monete. I valori da `C[0]` a `C[5]` devono essere gli indici delle monete (quindi numeri da $1$ a $6$) nell'ordine dalla moneta più leggera a quella più pesante.
  - Il tuo programma dovrebbe chiamare questa funzione solo da `orderCoins()`, e solo una volta per test case.
  - Questa funzione non ha valore di ritorno.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — corrispondono alle impostazioni $1$, $2$ e $3$ rispettivamente.
  - `A`, `B`, `C`: Le monete che sono poste nei piatti $A$, $B$, e $C$ rispettivamente. `A`, `B`, e `C` devono essere tre interi distinti compresi tra $1$ e $6$.
  - Ciascuna di queste funzioni restituisce uno dei numeri `A`, `B`, e `C`: l'indice della moneta appropriata. Per esempio, `getHeaviest(A, B, C)` restituisce il numero della moneta più pesante tra le tre fornite.
- `getNextLightest(A, B, C, D)` — corrisponde all'impostazione $4$.
  - `A`, `B`, `C`, `D`: Le monete che sono poste nei piatti $A$, $B$, $C$, e $D$ rispettivamente. `A`, `B`, `C`, e `D` devono essere quattro interi distinti compresi tra $1$ e $6$.
  - La funzione restituisce uno dei numeri `A`, `B`, e `C`: il numero della moneta selezionata dalla bilancia come descritto sopra per l'impostazione $4$. Cioè, la moneta restituita è la più leggere tra le monete nei piatti $A$, $B$, e $C$ che sono più pesanti della moneta nel piatto $D$; oppure se non ve ne sono è la più leggera di quelle nei tre piatti $A$, $B$, e $C$.

# Punteggio

Non ci sono subtask in questo problema. Invece, il tuo punteggio verrà calcolato sulla base del numero di pesate (numero totale di chiamate alle funzioni `getLightest()`, `getHeaviest()`, `getMedian()` e/o `getNextLightest()`) effettuate.

Il tuo programma verrà eseguito più volte, con diversi test case per ciascuna esecuzione. Sia $r$ il numero di esecuzioni totale (fissato dal dataset di test). Se il tuo programma *in un qualunque test case di una qualunque esecuzione* ordina le monete non correttamente, riceverà $0$ punti totali. Se invece il tuo programma ordina sempre correttamente le monete, ogni esecuzione è valutata individualmente come segue.

Sia $Q$ il più piccolo numero per cui è possibile ordinare una qualunque sequenza di sei monete utilizzando $Q$ pesate sulla bilancia di Amina. Per rendere il problema più interessante, non riveliamo il valore di $Q$.

Supponiamo che il massimo numero di pesate tra tutti i test case di tutte le esecuzioni sia $Q + y$ per un qualche intero positivo $y$, e consideriamo una singola esecuzione del programma. Supponiamo che invece il massimo numero di pesate tra tutti i test case di *questa* esecuzione sia $Q + x$ per un qualche intero positivo $x$ (se utilizzi meno di $Q$ pesate per ogni test case, si considera $x = 0$). Quindi, il punteggio di questa esecuzione sarà $\frac{100}{r((x+y)/5+1)}$, arrotondato *per difetto* a due cifre decimali dopo la virgola. In particolare, se il tuo programma fa al massimo $Q$ pesate in ogni test case di ogni esecuzione, prenderà $100$ punti.

# Esempio

Supponiamo che le monete siano ordinate $3\ 4\ 6\ 2\ 1\ 5$ dalla più leggera alla più pesante.

| Chiamata | Return | Spiegazione |
|---|---|---|
| getMedian(4, 5, 6) | 6 | La moneta $6$ è mediana tra le monete $4$, $5$, e $6$. |
| getHeaviest(3, 1, 2) | 1 | La moneta $1$ è la più pesante tra le monete $1$, $2$, e $3$. |
| getNextLightest(2, 3, 4, 5) | 3 | Le monete $2$, $3$, e $4$ sono tutte più leggere della moneta $5$, quindi la più leggera tra esse ($3$) è restituita. |
| getNextLightest(1, 6, 3, 4) | 6 | Le monete $1$ e $6$ sono entrambe più pesanti della moneta $4$. Tra di esse, la moneta $6$ è la più leggera. |
| getHeaviest(3, 5, 6) | 5 | La moneta $5$ è la più pesante tra le monete $3$, $5$ e $6$. |
| getMedian(1, 5, 6) | 1 | La moneta $1$ è mediana tra le monete $1$, $5$ e $6$. |
| getMedian(2, 4, 6) | 6 | La moneta $6$ è median tra le monete $2$, $4$ e $6$. |
| answer([3, 4, 6, 2, 1, 5]) | | Il programma ha trovato la risposta corretta per questo test case. |

# Grader di prova

Il grader di prova legge l'input nel seguente formato:

- riga $1$: $T$ —- il numero di test case
- righe da $2$ a $T + 1$: una sequenza di $6$ numeri distinti compresi tra $1$ e $6$, l'ordine delle monete dalla più leggera alla più pesante.

Per esempio, un input che consiste di due test case in cui le monete sono ordinate $1\ 2\ 3\ 4\ 5\ 6$ e $3\ 4\ 6\ 2\ 1\ 5$ è il seguente:

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

Il grader di prova stampa l'array che è stato passato come parametro alla funzione `answer()`.