Scales

Amina are șase monezi numerotate de la 1 la 6. Ea știe că monezile au greutăți diferite. Ea ar dori să le ordoneze în raport cu greutatea. Pentru acest scop ea a construit un nou tip de balanță.

O balanță traditională are două talere. Pentru a utiliza o astfel de balanță, se așează câte o monedă pe fiecare taler și balanța determină care dintre monezi este mai grea.
Noua balanță a Aminei este mult mai complexă. Aceasta are patru talere, etichetate A, B, C și D. Balanța are patru setari diferite, fiecare răspunzând unei anumite întrebari în legatură cu monezile. Pentru a utiliza balanța, Amina trebuie să plaseze exact câte o monedă pe fiecare dintre talerele A, B și C. În plus, pentru a patra setare ea trebuie sa așeze deasemenea exact o monedă pe talerul D.

Cele patru setari vor răspunde la următoarele patru întrebări:

1. Care dintre monezile de pe talerele A, B și C este cea mai grea?
2. Care dintre monezile de pe talerele A, B și C este cea mai ușoară?
3. Care dintre monezile de pe talerele A, B și C este cea mediană? (Adică nu este nici cea mai grea nici cea mai ușoară dintre cele trei.)
4. Dintre monezile de pe talerele A, B și C se consideră doar cele care sunt mai grele decât cea situată pe talerul D. Dacă există astfel de monezi , care dintre aceste monezi este cea mai usoară? Altfel, dacă nu există o astfel de monedă , care dintre monezile situate pe talerele A, B și C este cea mai usoară?

Cerință

Scrie un program care ordonează cele șase monezi ale Aminei în funcție de greutate. Programul poate să ceară balanței Aminei să compare greutăți ale monezilor. Programului tău i se vor da câteva teste spre rezolvare, fiecare corespunzând unui nou set de șase monede.

Programul tău trebuie sa implementeze funcțiile `init` și `orderCoins`. Pe parcursul fiecărei rulări, grader-ul va apela inițial functia `init` exact o dată. Aceasta va furniza numărul de teste și vă va permite sa inițializați orice variabilă. Apoi grader-ul va apela functia `orderCoins()` câte o dată pentru fiecare test.

- `init(T)`
  - `T`: Numărul de teste pe care programul tău va trebui să îl rezolve la rularea curentă. `T` este un întreg din intervalul $1, \ldots, 18$.
  - Aceasată funcție nu va returna nicio valoare.
- `orderCoins()`
  - Această funcție va fi apelată exact o dată pentru fiecare test.

- Funcția trebuie să determine corect ordinea monezilor Aminei apelând funcțiile grader-ului `getHeaviest()`, `getLightest()`, `getMedian()`, și/sau `getNextLightest()`.
- În momentul în care funcția cunoaște ordinea corectă, va trebui sa raporteze acest lucru prin apelul functiei grader-ului `answer()`.
- După apelul `answer()`, funcția `orderCoins()` trebuie sa returneze. Ea nu va returna nicio valoare.

Poți utiliza urmatoarele funcții ale grader-ului in programul tău:

- `answer(W)` — programul trebuie să utilizeze această funcție pentru a raporta răspunsul pe care l-a găsit.
  - `W`: Un șir de lungime 6 conținând ordinea corectă a monezilor. `W[0]` până la `W[5]` trebuie să fie numerele monezilor (adică numere de la 1 la 6) în ordine de la cea mai usoară la cea mai grea monedă.
  - Programul tău trebuie să apeleze această funcție din `orderCoins()`, o dată pentru fiecare test.
  - Această funcție nu returnează nicio valoare.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — Aceastea corespund setărilor 1, 2, respectiv 3 pentru balanța Aminei.
  - `A`, `B`, `C`: Monezile care sunt puse pe talerele A, B, respectiv C. `A`, `B`, și `C` trebuie să fie trei întregi distincți, fiecare între 1 și 6 inclusiv.
  - Fiecare funcție returnează unul dintre numerele `A`, `B` și `C`: numărul asociat monezii corecte. De exemplu , `getHeaviest(A, B, C)` returnează numărul celei mai grele dintre cele trei monezi date.
- `getNextLightest(A, B, C, D)` — aceasta corespunde setării 4 pentru balanța Aminei
  - `A`, `B`, `C`, `D`: Monezile care sunt puse pe talerele A, B, C, respectiv D. `A`, `B`, `C` și `D` trebuie să fie patru întregi distincți, fiecare între 1 și 6 inclusiv.
  - Funcția returnează unul dintre numerele `A`, `B` și `C`: numărul monezii selectate de balanță așa cum e descrisă anterior setarea 4. Adică, moneda returnată este cea mai ușoară dintre monezile de pe talerele A, B și C care este mai grea decât cea de pe talerul D; sau dacă niciuna nu este mai grea decât cea de pe talerul D, moneda returnată este pur și simplu cea mai usoară dintre monezile situate pe talerele A, B și C.

Punctaj

Această problemă nu are subprobleme. În schimb scorul tău va fi calculat în funcție de numărul de cântăriri (adică de numărul total de apeluri ale funcțiilor grader `getLightest()`, `getHeaviest()`, `getMedian()` și/sau `getNextLightest()`) pe care le face programul tău.

Programul tău va fi rulat de mai multe ori și pe mai multe teste la fiecare rulare. Fie r numărul de rulări ale programului tău. Acest număr este fixat în datele de test. Dacă programul tău nu ordonează monezile corect la oricare test din oricare rulare, vei obține 0 puncte. Altfel, fiecare rulare va fi punctată individual după cum urmează.

Fie Q cel mai mic număr posibil de cântîriri astfel încât să se poată sorta orice șir de șase monezi folosind Q cântăriri cu balanța Aminei. Pentru a face problema mai interesantă nu vom dezvălui aici valoarea lui Q.

Să presupunem că cel mai mare număr de cântăriri pe toate testele din toate rulările este $Q + y$ unde y este un întreg. Să considerăm apoi o singură rulare a programului tău. Fie $Q + x$ cel mai mare număr de cântîriri dintre toate cele T teste - unde x este un întreg nenegativ. (Dacă utilizați mai putin de Q cântăriri pe fiecare test atunci $x = 0$.) Scorul rulării curente va fi $\frac{100}{r((x+y)/5+1)}$, rotunjit *în jos* la două zecimale.

În particular, dacă programul face cel mult Q cântăriri pe fiecare test vei obține 100 de puncte.

Exemplu

Presupunem că monezile sunt ordonate **3 4 6 2 1 5** de la cea mai ușoară la cea mai grea.

Apel de funcție	Returnări	Explicație
getMedian(4, 5, 6)	6	Moneda **6** este cea mediană între monezile **4**, **5** și **6**.
getHeaviest(3, 1, 2)	1	Moneda **1** este cea mai grea dintre monezile **1**, **2** și **3**.
getNextLightest(2, 3, 4, 5)	3	Monezile **2**, **3** și **4** sunt toate mai ușoare decât moneda **5**, deci cea mai ușoară dintre ele (**3**) este returnată.
getNextLightest(1, 6, 3, 4)	6	Monezile **1** și **6** sunt ambele mai grele decât moneda **4**. Dintre monezile **1** și **6**, moneda **6** este cea mai ușoară.
getHeaviest(3, 5, 6)	5	Moneda **5** este cea mediană între monezile **3**, **5** și **6**.
getMedian(1, 5, 6)	1	Moneda **1** este cea mediană între monezile **1**, **5** și **6**.
getMedian(2, 4, 6)	6	Moneda **6** este cea mediană între monezile **2**, **4** și **6**.
answer([3, 4, 6, 2, 1, 5])		Programul a găsit răspunsul corect pentru acest test.

Grader-ul de pe calculatorul tău

Grader-ul de pe calculatorul tău citește datele de intrare în următorul format:

- linia **1**: T —- numărul de teste
- fiecare din liniile de la **2** la $T + 1$: un șir de **6** numere distincte de la **1** la **6**: ordinea monezilor de la cea mai ușoară la cea mai grea.

De exemplu, dacă datele de intrare constau în două teste unde monezile sunt ordonate **1 2 3 4 5 6** și **3 4 6 2 1 5** formatul datelor de intrare este următorul:

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

Grader-ul de pe calculatorul vostru afișează șirul trimis ca parametru de funcția `answer()`.