Mérések

Amina-nak hat különböző súlyú érméje van, 1-től 6-ig sorszámozva. Növekvő sorrendbe szeretné rendezni az érméket súly szerint.
A feladat elvégzéséhez van egy speciális mérőeszköze, ami különbözik a hagyományos mérlegektől. Négy helyre lehet tárgyat rakni, a helyek neve: A, B, C és D. Az eszköz négy különböző beállítással mérhet, mindegyik beállítás bizonyos kérdésre ad választ.
Az A, B és C helyre Amina pontosan egy érmét tehet. Csak a negyedik beállításban kell a D helyre pontosan egy érmét tenni.

A négy beállítás a következő kérdésekre ad választ:

- Az 1. megadja A, B és C közül a legnehezebbet.
- A 2. megadja A, B és C közül a legkönnyebbet.
- A 3. megadja A, B és C közül a középsőt.
- A 4.-ben vegyük A, B és C közül azokat, amelyek nehezebbek, mint D! Ha van ilyen, akkor megadja közülük a legkönnyebbet; egyébként az A, B és C közül a legkönnyebbet adja.

Feladat

Írj programot, amely növekvő sorrendbe rendezi Amina érméit! A program a speciális mérőeszközt használhatja a sorrend meghatározására.
A programnak egy futással több érme hatost kell rendeznie!
Az `init` és az `orderCoins` függvényeket kell megvalósítanod! Az értékelő először az `init` függvényt hívja, pontosan egyszer. Ez megkapja a futással rendezendő hatosok számát, továbbá itt inicializálhatsz változókat. Az értékelő minden hatosra egyszer hívja az `orderCoins()` függvényt.

- `init(T)`
  - `T`: A rendezendő hatosok száma, értéke 1 és 18 közötti.
  - Nincs visszadott értéke.
- `orderCoins()`
  - Minden hatosra egyszer hívják.
  - Meg kell adnia az érmék helyes sorrendjét a `getLightest()`, `getHeaviest()`, `getMedian()` és/vagy `getNextLightest()` függvényeket használva, amelyek a négy mérési módot valósítják meg.
  - A helyes eredmény ismeretében az `answer()` függvényt kell meghívni, ezzel kell közölni a rendezett sorrendet!

- Az `answer()` hívása után az `orderCoins()` függvénynek be kell fejeződnie, visszaadott érték nélkül!

Az alábbi függvényeket kell használnod:

- `answer(C)` — ezzel kell közölni a rendezett sorrendet.
  - `C`: 6 elemű tömb, amely az érmék sorrendjét tartalmazza `C[0]`-tól `C[5]`-ig érme sorszámokat (azaz 1 és 6 közötti számokat) súly szerint növekvő sorrendben.
  - Az `orderCoins()` függvényből kell hívni, pontosan egyszer!
  - Nincs visszadott értéke.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — az 1, 2 és 3 beállítások megvalósításai.
  - `A`, `B`, `C`: Az A, B és C helyekre tett érmék sorszámai, 1 és 6 közötti különböző érme sorszámok.
  - Mindegyik az `A`, `B` és `C` valamelyikét adja a beállításnak megfelelően: Például `getHeaviest(A, B, C)` megadja a legnehezebb érme sorszámát a három közül
- `getNextLightest(A, B, C, D)` — A negyedik beállítás megvalósítása.
  - `A`, `B`, `C`, `D`: Az A, B, C és D helyekre tett érmék sorszámai, 1 és 6 közötti különböző érme sorszámok.
  - Visszaadott értéke `A`, `B` és `C` valamelyike a negyedik beállításnak megfelelően. Vegyük `A`, `B` és `C` közül azokat, amelyek nehezebbek, mint `D`! Ha van ilyen, akkor megadja közülük a legkönnyebbet; egyébként az `A`, `B` és `C` közül a legkönnyebbet adja.

Pontozás

Nincsenek részfeladatok, pontszámod a végrehajtott `getLightest()`, `getHeaviest()`, `getMedian()` és/vagy `getNextLightest()` hívások számától függ.

A tesztelés során a programodat r-szer hajtják végre. Hibás sorrendre 0 pontot kapsz. Ha egy sorrend jó, akkor a következőképpen számolják a pontszámodat.

Legyen Q a legkevesebb mérésszám, amivel bármilyen hatosra a helyes sorrend meghatározható! Ezt a számot nem adjuk meg.

Legyen $Q + y$ a programod összes futása összes sorbarendezésére a mérések maximális száma! Ha egy futásra az összes sorbarendezésre a mérések maximális száma $Q + x$, akkor az erre a futásra adott pontszám $\frac{100}{r((x+y)/5+1)}$ lesz, két tizedesjegyre lefelé kerekítve. Ha Q-nál kevesebbet mértél, akkor $x = 0$.

Ha a pogramod mindig legfeljebb Q mérést végez, akkor a maximális 100 pontot kapod.

Példa

Legyen $3\ 4\ 6\ 2\ 1\ 5$ a növekvő sorrend!

Függvényhívás	Visszaadott érték	Magyarázat
getMedian(4, 5, 6)	6	A 6 a középső a 4, 5 és 6 közül.
getHeaviest(3, 1, 2)	1	Az 1 a legnehezebb az 1, 2 és 3 közül.
getNextLightest(2, 3, 4, 5)	3	A 2, 3 és 4 mindegyike könnyebb az 5-nél, közülük a legkönnyebb (3) a visszaadott érték.
getNextLightest(1, 6, 3, 4)	6	Az 1 és 6 nehezebb, mint a 4. Az 1 és a 6 közül a 6 a könnyebb.
getHeaviest(3, 5, 6)	5	Az 5 a legnehezebb a 3, 5 és 6 közül.
getMedian(1, 5, 6)	1	Az 1 a középső az 1, 5 és 6 közül.
getMedian(2, 4, 6)	6	A 6 a középső a 2, 4 és 6 közül.
answer([3, 4, 6, 2, 1, 5])		A program megtalálta a helyes sorrendet.

Minta értékelő

Az alábbi formában olvas a bemenetről:

- Az 1. sor: T —- a rendezendő hatosok száma
- A $2 \ldots T+1$. sorok: 6 darab különböző 1 és 6 közötti számot tartalmaznak, a helyes rendezett sorrendet.

Például, ha bemenet ezt a két hatost tartalmazza $1\ 2\ 3\ 4\ 5\ 6$ és $3\ 4\ 6\ 2\ 1\ 5$, akkor a bemenet:

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

Az értékelő kiírja az `answer()` függvény paraméterében kapott tömböt.