

Ваги

Перо има 6 монети, нумерирани со целите броеви од **1** до **6**, и знае дека сите имаат различна тежина. Тој сака да ги подреди според тежината. На располагање има нов вид на вага со тасови.

Обичната вага има 2 таса. Кога таа се користи, се става по една монета на секој тас и вагата ќе утврди која монета е потешка.

Новата вага е посложена. Таа има 4 таса, означени со **A**, **B**, **C** и **D**. Вагата има 4 режими на работа, при што секој режим дава одговор на различно прашање во врска со монетите. За да ја користи вагата, Перо мора да стави по точно една монета на секој од тасовите **A**, **B** и **C**. Во четвртиот режим се става точно една монета и на тасот **D**.

Различните режими одговараат на следново:

1. Која од монетите ставени на тасовите **A**, **B** и **C** е најтешката?
2. Која од монетите ставени на тасовите **A**, **B** и **C** е најлесната?
3. Која од монетите ставени на тасовите **A**, **B** и **C** е средната? (Монетата која не е ниту најлесна ниту најтешка)
4. Помеѓу монетите на тасовите **A**, **B** и **C**, земи ги предвид само тие кои се потешки од таа на тасот **D**. Ако има такви монети, која од нив е најлесната? Инаку, ако нема такви, која од монетите на тасовите **A**, **B** и **C** е најлесната?

Задача

Напишете програма која ќе ги подреди монетите на Перо според тежината. Програмата може да ја прашува вагата на Перо, за да ги споредува тежините на монетите. На вашата програма ќе и биде дадено да реши неколку тест случаи, а секој од нив ќе се однесува на ново множество од 6 монети.

Вашата програма треба да ги имплементира функциите `init` и `orderCoins`. При секое извршување на вашата програма, оценувачот прво, точно еднаш, ќе ја повика `init`. Ова ви го дава бројот на тест случаи и ви овозможува да иницијализирате некои ваши променливи. Оценувачот потоа ќе ја повика функцијата `orderCoins()`, по еднаш за секој тест случај.

- `init(T)`

- **T**: Бројот на тест случаи што ќе треба да ги реши вашата програма во текот на ова извршување. **T** е цел број од рангот **1, ..., 18**.
- Оваа функција не враќа никаква вредност.

- `orderCoins()`

- Оваа функција се повикува точно по еднаш за секој тест случај.
- Функцијата треба да го определи точниот редослед на монетите на Перо со повикување на функциите од оценувачот `getHeaviest()`, `getLightest()`, `getMedian()`, и/или `getNextLightest()`.
- Откако функцијата ќе го дознае точниот редослед, ќе треба да го достави истиот со повикување на функцијата `answer()` од оценувачот.
- По повикот на `answer()`, функцијата `orderCoins()` треба да го заврши своето извршување. Таа не враќа никаква вредност.

Може да ги користите следните функции од оценувачот во вашата програма:

- `answer(W)` — вашата програма треба да ја користи оваа функција за да го достави одговорот кој што го пронашла.
 - `W`: низа со должина 6 која што го содржи точниот редослед на монетите. `W[0]` до `W[5]` треба да бидат броевите на монетите (т.е., броеви од **1** до **6**), во редослед од најлесната кон најтешката монета.
 - Вашата програма треба да ја повикува оваа функција само од `orderCoins()`, по еднаш за секој тест случај.
 - Оваа функција не враќа никаква вредност.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — овие одговараат на режимите на работа 1, 2 и 3, соодветно, на вагата на Перо.
 - `A, B, C`: Монетите што се ставени на тасовите **A**, **B** и **C**, соодветно. `A, B` и `C` треба да бидат три различни цели броеви, секој помеѓу **1** и **6**.
 - Секоја функција враќа еден од броевите `A, B` и `C`: бројот на соодветната монета. На пример, `getHeaviest(A, B, C)` го враќа бројот на најтешката од трите дадени монети.
- `getNextLightest(A, B, C, D)` — оваа функција одговара на режимот на работа 4 за вагата на Перо.
 - `A, B, C, D`: Монетите што се ставени на тасовите **A**, **B**, **C** и **D**, соодветно. `A, B, C` и `D` треба да бидат четири различни цели броеви, секој помеѓу **1** и **6**.
 - Функцијата враќа еден од броевите `A, B` и `C`: бројот на монетата избрана од страна на вагата, како што е опишано погоре за режимот на работа 4. Со други зборови, вратената монета е најлесната помеѓу оние монети на тасовите **A**, **B** и **C** кои што се потешки од монетата на тасот **D**; или, ако ниту една од нив не е потешка од монетата на тасот **D**, вратената монета е едноставно најлесната од трите монети на тасовите **A**, **B** и **C**.

Оценување

Нема подзадачи за оваа задача. Наместо тоа, резултатот ќе се базира на тоа колку мерења (вкупен број на повици на функциите на грејдерот `getLightest()`, `getHeaviest()`, `getMedian()` и/или `getNextLightest()`) ќе направи вашата програма.

Вашата програма ќе биде пуштена (извршена) неколку пати. Секој пат на по неколку тест случаи. Нека r е бројот на извршувања на вашата програма. Овој број е утврден според множеството на подготвени тестни податоци. Ако вашата програма НЕ ГИ ПОДРЕДИ ВО ВИСТИНСКИ РЕДОСЛЕД монетите, во кој било тест случај, од кое било извршување, ќе биде оценета со 0 поени. Ако даде точно подредување за сите случаи, ќе се оцени според следниот принцип:

Нека Q е најмалиот број на мерења (со скалата на Перо) со кои може да се подреди низа од 6 монети. Таков број постои, но, за да ја направиме задачата поинтересна, овде не ја откриваме вредноста за Q .

Претпоставете дека во некој од тест случаите за сите извршувања на програмата сте направиле најмногу мерења и тоа точно $Q + y$ за некој цел број y .

Да разгледаме сега едно извршување на програмата. Нека во некој тест случај, од сите T тест случаи од тоа извршување на програмата, сте направиле најмногу мерења - на пример $Q + x$ за некој ненегативен цел број x . (Ако користите помалку од Q мерења за секој тест случај, тогаш $x = 0$.)

Е сега, резултатот за ова извршување ќе биде $\frac{100}{r((x+y)/5+1)}$, *надолно* заокружено на 2 цифри после децималната запирка.

Конкретно, ако вашата програма направи најмногу Q мерења во секој тест случај во секое извршување, ќе добиете 100 поени.

Пример

Да претпоставиме дека монетите се подредени **3 4 6 2 1 5** во редослед од најлесната кон најтешката.

Функциски повик	Враќа	Објаснување
getMedian(4, 5, 6)	6	Монетата 6 е средната помеѓу монетите 4, 5 и 6.
getHeaviest(3, 1, 2)	1	Монетата 1 е најтешката помеѓу монетите 1, 2 и 3.
getNextLightest(2, 3, 4, 5)	3	Монетите 2, 3 и 4 се сите полесни од монетата 5, па најлесната помеѓу нив (3) е вратена како резултат.
getNextLightest(1, 6, 3, 4)	6	Монетите 1 и 6 се двете потешки од монетата 4. Помеѓу монетите 1 и 6, монетата 6 е најлесната.
getHeaviest(3, 5, 6)	5	Монетата 5 е најтешката помеѓу монетите 3, 5 и 6.
getMedian(1, 5, 6)	1	Монетата 1 е средната помеѓу монетите 1, 5 и 6.
getMedian(2, 4, 6)	6	Монетата 6 е средната помеѓу монетите 2, 4 и 6.
answer([3, 4, 6, 2, 1, 5])		Програмата го пронајде точниот одговор за овој тест случај.

Пример оценувач

Пример оценувачот чита влезни податоци во следниот формат:

- линија 1: T — бројот на тест случаи

- секоја од линиите од **2** до **$T + 1$** : секвенца од **6** различни броеви од **1** до **6**: редоследот на монетите од најлесната кон најтешката.

На пример, влез кој се состои од два тест случаи каде монетите се подредени **1 2 3 4 5 6** и **3 4 6 2 1 5** изгледа вака:

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

Пример оценувачот (грејдерот) ја печати низата која беше предадена како параметар на функцијата `answer()`.