

Scales

Амина има 6 монети, номерирани от **1** до **6**, с различни тегла и иска да ги подреди по големина на теглото в нарастващ ред. За целта тя разработила нов тип везна.

Традиционната везна има две блюда и показва коя от двете монети, поставени на всяко едно от блюдата е по-тежка. Везната на Амина е много по-сложна. Тя има четири блюда, означени с **A**, **B**, **C** и **D** и четири режима на използване, при всеки от които показва отговор на специфичен въпрос. За всеки от първите три режима Амина трябва да постави точно по една монета на блюдата **A**, **B** и **C**, а за четвъртия режим — точно една монета и в блюдото **D**.

Везната дава отговор на следните въпроси:

1. Коя от монетите в блюдата **A**, **B** и **C** е най-тежка?
2. Коя от монетите в блюдата **A**, **B** и **C** е най-лека?
3. Коя от монетите в блюдата **A**, **B** и **C** е със средно тегло?
4. Разглеждаме само монетите в блюдата **A**, **B** и **C**, които са по-тежки от монетата в **D**. Ако има такива монети, коя от тях е най-лека, а ако няма — коя от трите монети в **A**, **B** и **C** е най-лека?

Задача

Напишете програма, която да подрежда монетите по големина на теглото, като задава последователност от допустими въпроси. Програмата трябва да е в състояние да извърши няколко сортирания при едно изпълнение, с различни комбинации от тегла на шестте монети.

Програмата трябва да имплементира функциите `init()` и `orderCoins()`. При всяко изпълнение на програмата, гредърът първо ще извика еднократно функцията `init()`, за да зададе броя на сортиранията, които трябва да бъдат направени, и да позволи инициализирането на променливи. След това, грейдърът ще извика функцията `orderCoins()` толкова пъти, колкото е броят на сортиранията.

- `init(T)`
 - **T**: брой на сортиранията, които програмата трябва да извърши при едно изпълнение — цяло число в интервала от **1** до **18**.
 - Функцията не трябва да връща стойност.
- `orderCoins()`
 - Тази функция ще бъде извикана по един път за всяко от исканите **T** сортирания.
 - Функцията трябва да определи правилното подреждане на монетите с помощта на функциите `getHeaviest()`, `getLightest()`, `getMedian()` и/или

`getNextLightest()`.

- Щом като намери търсеното подреждане, функцията трябва да го съобщи на грейдъра с извикване на функцията `answer()` и да завърши работа, без да връща стойност.

В програмата може да използвате следните функции на грейдъра:

- `answer(W)` - използвайте тази функция, за да съобщите на тестващата програма правилното подреждане на монетите.
 - W : масив с 6 елемента съдържащ правилното подреждане на монетите, т.е. $W[0], \dots, W[5]$ трябва да съдържат номерата на монетите (различни числа от **1** до **6**), в реда от най-леката до най-тежката.
 - Програмата трябва да извика тази функция от `orderCoins()` точно един път.
 - Тази функция не връща стойност.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — отговарят на въпросите **1**, **2** и **3**, съответно.
 - A, B, C : номерата на монетите поставени в блюдата **A**, **B** и **C** съответно — три различни цели в интервала от **1** до **6** включително.
 - Всяка от трите функции връща номера на монетата, която е отговор на съответния въпрос. Например, `getHeaviest(A, B, C)` връща номера на най-тежката монета, поставена в едно от блюдата **A**, **B** и **C**.
- `getNextLightest()` — отговаря на въпрос 4.
 - A, B, C, D : номерата на монетите поставени в блюдата **A**, **B**, **C** и **D**, съответно — четири различни цели в интервала от **1** до **6** включително.
 - Функцията връща номера на тази от монетите, поставени в блюдата **A**, **B** и **C**, която е отговор на въпрос 4.

Оценяване

Тази задача няма подзадачи. Оценяването ще се извърши на база на броя на извикванията на някои от функциите `getHeaviest()`, `getLightest()`, `getMedian()` и/или `getNextLightest()`.

Програмата ще бъде изпълнена много пъти, като всеки път трябва да направи по няколко сортирания. Нека r е броят на извикванията на вашата програма от грейдъра — число фиксирано в множеството от тестове. Ако програмата направи поне едно грешно сортиране при някое от изпълненията, оценката ще бъде **0**. В противен случай, всяко извикване се оценява индивидуално както следва.

Нека Q е най-малкото число такова, че е възможно да се сортират кои да са шест монети с не повече от Q претегляния с везната на Амина (за да е по-интересно, няма да знаете това число).

Да предположим, че максималният брой претегляния, които програмата прави за кое да е

сортиране, при кое да е изпълнение, е $Q + y$ за някое цяло число y . Разгледайте кое да е изпълнение на програмата и нека най-големият брой на претеглянията направени от програмата ви за някое от сортиранията при това изпълнение е $Q + x$ за някое неотрицателно цяло x (ако програмата е направила по-малко от Q претегляния, тогава x е нула). Тогава, за това изпълнение на програмата ще получите като оценка закръгленото *надолу* до две цифри след десетичната точка число $\frac{100}{r((x+y)/5+1)}$. В частност, ако на всяко сортиране, при всяко изпълнение, програмата ви прави не-повече от Q претегляния, резултатът ви ще бъде 100 точки.

Пример

Да предположим, че правилният ред на монетите е **3 4 6 2 1 5** от най-леката към най-тежката.

Function call	Returns	Explanation
getMedian(4, 5, 6)	6	Монетата 6 е средната измежду монетите 4 , 5 , и 6 .
getHeaviest(3, 1, 2)	1	Монетата 1 е най-тежката измежду монетите 1 , 2 , and 3 .
getNextLightest(2, 3, 4, 5)	3	Монетите 2 , 3 , и 4 са по-леки от монетата 5 , така че функцията връща най-леката (3).
getNextLightest(1, 6, 3, 4)	6	Монетите 1 и 6 са по-тежки от 4 . Измежду монетите 1 и 6 , монетата 6 най-леката.
getHeaviest(3, 5, 6)	5	Монетата 5 е най-тежката измежду монетите 3 , 5 и 6 .
getMedian(1, 5, 6)	1	Монетата 1 е средна измежду 1 , 5 и 6 .
getMedian(2, 4, 6)	6	Монетата 6 е средната измежду монетите 2 , 4 и 6 .
answer([3, 4, 6, 2, 1, 5])		Програмата намери вярното подреждане.

Примерен грейдър

Примерният грейдер приема вход във вида:

- ред 1: брой на сортиранията
- всеки от редовете от **2** до $T + 1$ — редица от **6** различни числа в интервала от **1** до **6**, които представляват поредните сортирани шест монети, от най-леката към най тежката.

Например

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

Примерният грейдер отпечатава масива, зададен от програмата, като аргумент на функцията `answer()`.