

Взвешивания

У Амины есть 6 монет, пронумерованных от **1** до **6**. Она знает, что все монеты имеют разный вес. Амина хочет упорядочить монеты по весу. Для этого у нее есть специальные весы.

У классических весов есть две чаши. Для того, чтобы воспользоваться ими, необходимо положить по одной монете на каждую чашу весов, и весы определят, какая из монет тяжелее.

Новые весы Амины устроены сложнее. У них есть четыре чаши, обозначенные **A**, **B**, **C** и **D**. Эти весы работают в четырех режимах, в каждом из которых они отвечают на различные вопросы о положенных на них монетах. В первых трех режимах Амина должна положить ровно по одной монете на каждую из чаш **A**, **B** и **C**. При использовании четвертого режима она должна, кроме чаш **A**, **B** и **C**, дополнительно положить ровно одну монету на чашу **D**.

Четыре режима отвечают на следующие четыре вопроса соответственно:

1. Какая из монет на чашах **A**, **B**, **C** самая тяжелая?
2. Какая из монет на чашах **A**, **B**, **C** самая легкая?
3. Какая из монет на чашах **A**, **B**, **C** средняя по весу?
4. Среди монет на чашах **A**, **B**, **C**, рассматриваются только монеты, которые тяжелее монеты на чаше **D**. Если такие монеты есть, то весы сообщают, какая из них самая легкая. Иначе, если таких монет нет, то весы сообщают, какая из монет на чашах **A**, **B**, **C** самая легкая.

Постановка задачи

Напишите программу, которая расположит шесть монет Амины по весу.

Программа может задавать вопросы весам Амины. Программа должна решить задачу для нескольких наборов входных данных, каждый из которых соответствует новому набору из шести монет.

Необходимо реализовать две функции `init` и `orderCoins`. Во время каждого запуска программы сначала ровно один раз будет вызвана функция `init`, которой сообщается количество наборов монет в этом тесте, а также может выполняться необходимая инициализация переменных. После этого будет вызвана функция `orderCoins()` один раз для каждого набора монет.

- `init(T)`
 - `T` — количество наборов монет, для которых необходимо решить задачу в этом тесте. `T` — целое число в промежутке от 1 до 18.
 - Эта функция не возвращает никакого значения.
- `orderCoins()`
 - Эта функция будет вызвана один раз для каждого набора монет.
 - Эта функция должна определить правильный порядок монет Амины, используя функции `getHeaviest()`, `getLightest()`, `getMedian()`, и/или `getNextLightest()`.
 - Когда удалось восстановить правильный порядок, необходимо вызвать функцию `answer()`.
 - После вызова функции `answer()`, функция `orderCoins()` должна завершиться. Эта функция не возвращает никакого значения.

В своей программе вы можете использовать следующие функции:

- `answer(W)` — эту функцию необходимо вызвать, чтобы сообщить найденный ответ.
 - `W` — массив из 6 элементов содержащий правильный порядок монет. Значения от `W[0]` до `W[5]` должны быть номерами монет, то есть, числами от **1** до **6** в порядке от самой легкой до самой тяжелой.
 - Эту функцию можно вызывать только один раз в каждом запуске функции `orderCoins()`.
 - Эта функция не возвращает никакого значения.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — эти функции соответствуют 1, 2 и 3 режимам работы весов Амины.
 - `A, B, C` — номера монет в чашах **A**, **B** и **C** соответственно. `A, B` и `C` должны быть тремя различными целыми числами от **1** до **6** включительно.
 - Каждая из функций возвращает одно из чисел `A, B, C`, соответствующее подходящей монете. Например, `getHeaviest(A, B, C)` возвращает номер самой тяжелой из трех переданных монет.
- `getNextLightest(A, B, C, D)` — эта функция соответствует четвертому режиму работы весов Амины
 - `A, B, C, D` — номера монет, в чашах **A**, **B**, **C**, **D** соответственно. `A, B, C` и `D` должны быть четырьмя различным целыми числами от **1** до **6** включительно.
 - Эта функция возвращает одно из чисел `A, B` или `C`, выбранное способом, описанным выше для четвертого режима работы. То есть, это номер самой легкой монеты на чашах **A**, **B**, **C**, которая тяжелее, чем монета на чаше **D**. Если ни одна из них не тяжелее монеты на чаше **D**, то возвращается самая легкая из монет на чашах **A**, **B**, **C**.

Система оценивания

В этой задаче нет подзадач. Баллы за решение зависят от количества взвешиваний, сделанных программой (суммарного количества вызовов функций `getLightest()`, `getHeaviest()`, `getMedian()` и/или `getNextLightest()`).

На каждом тесте программе необходимо решить задачу для нескольких наборов монет. Обозначим за r количество тестов в задаче. Это число зафиксировано тестовыми данными. Если решение неправильно восстановит порядок хотя бы для одного набора монет хотя бы в одном тесте, то оно будет оценено в 0 баллов. Иначе тесты оцениваются по отдельности.

Обозначим за Q минимальное количество взвешиваний, используя которое возможно отсортировать любой набор из шести монет с помощью весов Амины. Для усложнения задачи мы не сообщаем вам число Q . Пусть максимальное количество взвешиваний сделанных вашим решением на всех наборах монет во всех тестах равно $Q + y$ для некоторого целого числа y .

Пусть максимальное количество взвешиваний, сделанных вашим решением на всех наборах монет среди T наборов конкретного теста, равно $Q + x$ для некоторого целого числа x . (Если вы используете менее Q взвешиваний для всех наборов монет, то $x = 0$). В таком случае, количество баллов за этот тест будет равно $\frac{100}{r((x+y)/5+1)}$, округленному вниз до двух знаков после запятой.

В частности, если ваше решение делает не более Q взвешиваний на каждом наборе монет каждого теста, вы получите 100 баллов.

Пример

Пусть монеты упорядочены от легких к тяжелым как **3 4 6 2 1 5**.

Вызов функции	Возвращаемое значение	Пояснение
<code>getMedian(4,5,6)</code>	6	Монета 6 является средней по весу среди монет 4, 5 и 6.
<code>getHeaviest(3,1,2)</code>	1	Монета 1 самая тяжелая среди 1, 2, и 3.
<code>getNextLightest(2,3,4,5)</code>	3	Монеты 2, 3, и 4 легче, чем 5, по этому будет возвращена самая легкая из них (3).
<code>getNextLightest(1,6,3,4)</code>	6	Монеты 1 и 6 обе тяжелее, чем монета 4. Среди них, будет возвращена монета более легкая монета 6
<code>getHeaviest(3,5,6)</code>	5	Монета 5 самая тяжелая среди 3, 5 и 6.
<code>getMedian(1,5,6)</code>	1	Монета 1 средняя по весу, среди 1, 5 и 6.
<code>getMedian(2,4,6)</code>	6	Монета 6 средняя по весу, среди 2, 4 и 6.
<code>answer([3,4,6,2,1,5])</code>		Программа нашла корректный ответ для этого набора монет

Пример проверяющего модуля

Проверяющий модуль имеет следующий формат входных данных:

- Строка **1**: **T** — количество наборов монет
- На каждой из строк от **2** до **$T + 1$** : последовательность из **6** различных целых чисел от **1** до **6**: порядок монет от самой легкой, до самой тяжелой.

Например, если тест состоит из двух наборов монет, упорядоченных как **1 2 3 4 5 6** и **3 4 6 2 1 5**, то входные данные должны выглядеть следующим образом:

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

Проверяющий модуль выводит массив, который был передан функции `answer()`.