Терези

У Аміни є 6 монет, пронумерованих від 1 до 6. Вона знає, що всі монети мають різну вагу. Вона хоче впорядкувати їх відповідно до їх ваги. Для цього вона розробила новий вид терезів.

Традиційні терези мають дві шальки. Щоб скористатись такими терезами, ви кладете монету на кожну з шальок, а терези визначають яка з них важча.

Нові Амінині терези більш склідні. Вони мають 4 шальки, помічені A, B, C і D. Терези мають 4 різних налаштування, кожне з яких відповідає на різні запитання відносно ваги монет. Щоб використати терези, Аміна має покласти рівно по одній монеті на кожну із шальок A, B і C. Додатково, при четвертому налаштуванні вона також має положити рівно одну монету на шальку D.

Запитання, на які дають відповідь терези при кожному з чотирьох налаштувань:

1. Яка з монет на шальках A, B і C найважча?
2. Яка з монет на шальках A, B і C найлегша?
3. Яка з монет на шальках A, B і C є медіаною? (Це монета, яка не є ні найлегшою, ні найважчою з 3)
4. Серед монет на шальках A, B і C розглянемо лише монети, важчі ніж монета на шальці D. Якщо такі монети є, то яка з них є найлегшою? Інакше, якщо таких монет немає, яка з монет на шальках A, B і C є найлегшою?

Задача

Напишіть програму, яка впорядкує 6 монет Аміни у відповідності до їх ваги. Програма може викликати терези Аміни для порівняння ваги монет. Вашій програмі буде надано декілька наборів тестових даних для розв'язання, кожен з яких відповідає новому набору з 6 монет.

Ваша програма має реалізовувати функції `init` та `orderCoins`. Протягом кожного запуску вашої програми модуль перевірки спочатку рівно один раз буде викликати функцію `init`. Це надасть вам інформацію про кількість наборів тестових даних та дозволить проініціалізувати змінні. Після цього модуль перевірки буде викликати функцію `orderCoins()` один раз для кожного набору тестових даних.

- `init(T)`
  - `T`: Кількість наборів тестових даних, що має обробити ваша програма протягом цього запуску. `T` — ціле число з діапазону $1, \ldots, 18$.
  - Ця функція не повертає значення.
- `orderCoins()`

- Ця функція викликається рівно один раз для кожного набору тестових даних.
- Вона повинна визначити вірний порядок монет Аміни, викликаючи функції модуля перевірки `getHeaviest()`, `getLightest()`, `getMedian()` та `getNextLightest()`.
- Як тільки функція дізнається правильну відповідь, вона повинна сповістити про це модуль перевірки, викликавши функцію `answer()`.
- Після виклику `answer()`, функція `orderCoins()` повинна завершитись. Вона не повертає значення.

Ви можете використовувати в вашій програмі наступні функції модуля перевірки:

- `answer(W)` — ваша програма використовує цю функцію, щоб повідомити відповідь, яку вона знайшла.
  - `W`: Масив довжини 6, що містить правильний порядок монет. `W[0]` до `W[5]` мають бути номерами монет (тобто, числами від 1 до 6) в порядку від найлегшої до найважчої монети.
  - Ваша програма має викликати цю функцію лише з `orderCoins()`, лише один раз на кожен набір.
  - Ця функція не повертає значення.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — ці функції відповідають налаштуванням 1, 2 та 3 терезів Аміни, відповідно.
  - `A`, `B`, `C`: Монети, покладені на шальки A, B і C, відповідно. `A`, `B` і `C` мають бути трьома різними цілими числами, кожне з діапазону від 1 до 6 включно.
  - Кожна з функцій повертає одне із значень `A`, `B`, або `C`: номер відповідної монети. Наприклад, `getHeaviest(A, B, C)` повертає номер найважчої з трьох даних монет.
- `getNextLightest(A, B, C, D)` — відповідає налаштуванню 4 Амініних терезів.
  - `A`, `B`, `C`, `D`: Монети, покладені на шальки A, B, C та D, відповідно. `A`, `B`, `C` та `D` мають бути чотирма різними цілими числами, кожне з діапазону від 1 до 6 включно.
  - Функція повертає одне з чисел `A`, `B`, або `C`: номер монети, обраний терезами за правилом, наведеним вище для налаштування 4. Тобто, монета яку повернули найлегша серед монет, що лежать на шальках A, B і C і важчі за монету на шальці D; або, якщо жодна з них не важча за монету на шальці D, то повернута монета найлегша серед монет на шальках A, B і C.

Оцінювання

Ця задача не має підзадач. Замість цього, ваша оцінка буде залежати від кількості зважувань (загальна кількість викликів функцій `getLightest()`, `getHeaviest()`, `getMedian()` та `getNextLightest()`), що робить ваша програма.

Вашу програму буде виконано кілька разів з певною кількістю наборів тестових даних для кожного виконання. Нехай r — кількість запусків вашої програми. Цю кількість зафіксовано в

тестових даних. Якщо ваша програма вірно не впорядкує монети, принаймні в одному з тестових наборів при одному з запусків, вона отримає 0 балів. В іншому випадку кожен запуск оцінюється індивідуально наступним чином.

Нехай Q — найменше число, таке, що можливо відсортувати довільну послідовність із 6 монет використовуючи Q зважувань на терезах Аміни. Щоб зробити задачу цікавішою, ми не повідомляємо значення Q.

Припустимо, що найбіше число зважувань для всіх наборів тестів у всіх запусках — $Q + y$ для певного цілого y. Тоді розглянемо один із запусків програми. Нехай найбільша кількість зважувань серед усіх T наборів даних у цьому запуску — $Q + x$ для деякого невід'ємного цілого числа x. (Якщо ви використали менше ніж Q зважувань для кожного з тестових наборів, то $x = 0$.) Тоді, оцінкою за цей запуск буде $\frac{100}{r((x+y)/5+1)}$, заокруглене *вниз* до двох знаків після коми.

Зокрема, якщо ваша програма робить щонайбільше Q зважувань для кожного тестового набору даних при кожному зважуванні — ви отримаєте 100 балів.

Приклад

Нехай монети впорядковані наступним чином $3\ 4\ 6\ 2\ 1\ 5$ від найлегшої до найважчої.

Виклик функції	Результат	Пояснення
getMedian(4, 5, 6)	6	Монета 6 є медіаною серед монет 4, 5 та 6.
getHeaviest(3, 1, 2)	1	Монета 1 є найважчою серед монет 1, 2 та 3.
getNextLightest(2, 3, 4, 5)	3	Монети 2, 3 та 4 всі легші ніж 5, тому повертається найлегша з них (3).
getNextLightest(1, 6, 3, 4)	6	Монети 1 і 6 обидві важчі за 4. Серед монет 1 і 6, монета 6 є найлегшою.
getHeaviest(3, 5, 6)	5	Монета 5 є найважчою серед монет 3, 5 та 6.
getMedian(1, 5, 6)	1	Монета 1 є медіаною серед монет 1, 5 та 6.
getMedian(2, 4, 6)	6	Монета 6 є медіаною серед монет 2, 4 та 6.
answer([3, 4, 6, 2, 1, 5])		Програма знайшла привильну відповідь для цього набору тестових даних.

Приклад модуля перевірки

Отриманий вами модуль перевірки читає вхідні дані у наступному форматі:

- рядок 1: T — кількість наборів тестових даних
- кожен рядок з 2 по $T + 1$: послідовність з 6 різних чисел від 1 до 6: порядок монет від найлегшої до найважчої.

Наприклад, Вхідні дані, що складаються з двох наборів тестових даних, де монети впорядковані $1\ 2\ 3\ 4\ 5\ 6$ та $3\ 4\ 6\ 2\ 1\ 5$ виглядають так:

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

Отриманий модуль перевірки виводить масив, що було передано як параметр до функції `answer()`.