Chiếc Cân

Amina có sáu đồng xu, được đánh số thứ tự từ 1 đến 6. Cô biết là tất cả đồng xu có trọng lượng khác nhau. Cô muốn sắp xếp các đồng xu theo trọng lượng của chúng. Với mục đích này, cô đã thiết kế ra một loại cân mới.

Cân truyền thống có hai đĩa. Để sử dụng cân truyền thống, bạn đặt một đồng xu vào mỗi đĩa và chiếc cân sẽ xác định đồng xu nào nặng hơn.

Chiếc cân mới Anima thiết kế ra phức tạp hơn. Nó có bốn đĩa, được gán nhãn A, B, C, và D. Chiếc cân có bốn cấu hình sử dụng khác nhau, mỗi cấu hình trả lời một câu hỏi riêng về các đồng xu. Để sử dụng chiếc cân này, Anima phải đặt đúng một đồng xu vào từng đĩa A, B, và C. Đối với cấu hình thứ tư, cô còn phải đặt thêm đúng một đồng xu vào đĩa D.

Bốn cấu hình sẽ chỉ dẫn chiếc cân trả lời bốn câu hỏi sau:

1. Đồng xu nào trong số các đồng xu trên các đĩa A, B, và C có trọng lượng nặng nhất?
2. Đồng xu nào trong số các đồng xu trên các đĩa A, B, và C có trọng lượng nhẹ nhất?
3. Đồng xu nào trong số các đồng xu trên các đĩa A, B, và C có trọng lượng ở giữa? (Đây là đồng xu không phải nặng nhất cũng không phải nhẹ nhất trong ba đồng xu.)
4. Trong số các đồng xu trên các đĩa A, B, và C, chỉ xét các đồng xu có trọng lượng nặng hơn đồng xu trên đĩa D. Nếu có các đồng xu như vậy, đồng xu nào trong chúng có trọng lượng nhẹ nhất? Ngược lại, nếu không có những đồng xu như vậy, đồng xu nào trong số các đồng xu trên các đĩa A, B, và C có trọng lượng nhẹ nhất?

Nhiệm vụ

Hãy viết một chương trình để sắp xếp sáu đồng xu của Amina theo trọng lượng của chúng. Chương trình có thể sử dụng chiếc cân của Amina để so sánh trọng lượng các đồng xu. Chương trình của bạn sẽ phải giải các test cho trước, mỗi test tương ứng với một tập sáu đồng xu mới.

Chương trình của bạn phải cài đặt các hàm `init` và `orderCoins`. Trong mỗi lần chạy chương trình của bạn, chương trình chấm sẽ bắt đầu bằng việc gọi `init` đúng một lần. Việc này sẽ cho bạn biết số lượng test và cho phép bạn khởi tạo các biến. Sau đó, chương trình chấm sẽ gọi `orderCoins()` một lần đối với mỗi test.

- `init(T)`
  - `T`: Số lượng test mà chương trình của bạn phải giải cho lần chạy này. `T` là một số tự nhiên nằm trong khoảng $1, \ldots, 18$.
  - Hàm này không có giá trị trả về.
- `orderCoins()`

- Hàm này chỉ được gọi đúng một lần đối với mỗi test.
- Hàm này phải xác định thứ tự đúng cho các đồng xu của Amina bằng cách gọi các hàm của chương trình chấm `getHeaviest()`, `getLightest()`, `getMedian()`, và/hoặc `getNextLightest()`.
- Khi đã xác định được thứ tự đúng, hàm phải thông báo kết quả bằng cách gọi hàm `answer()` của chương trình chấm.
- Sau khi gọi `answer()`, hàm `orderCoins()` phải dừng. Hàm này không có giá trị trả về.

Trong chương trình của mình, bạn có thể sử dụng các hàm sau của chương trình chấm:

- `answer(W)` — Chương trình của bạn phải sử dụng hàm này để thông báo kết quả tìm được.
  - `W`: Mảng có 6 phần tử chứa thứ tự đúng của các đồng xu. `W[0]` đến `W[5]` phải là số thứ tự các đồng xu (nghĩa là các số từ 1 đến 6) theo thứ tự từ đồng xu có trọng lượng nhẹ nhất đến đồng xu có trọng lượng nặng nhất.
  - Chương trình của bạn chỉ được gọi hàm này từ `orderCoins()`, một lần đối với mỗi test.
  - Hàm này không có giá trị trả về.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — Các hàm này tương ứng theo đúng thứ tự với các cấu hình 1, 2 và 3 của chiếc cân của Amina.
  - `A`, `B`, `C`: Các đồng xu đặt theo đúng thứ tự trên các đĩa A, B, và C tương ứng. `A`, `B`, và `C` là ba số nguyên phân biệt, mỗi số có giá trị trong khoảng từ 1 đến 6 bao gồm cả hai giá trị đầu mút.
  - Mỗi hàm trả về một trong các số `A`, `B`, và `C`: Số thứ tự của đồng xu tương ứng. Ví dụ, `getHeaviest(A, B, C)` trả về số thứ tự của đồng xu nặng nhất trong ba đồng xu.
- `getNextLightest(A, B, C, D)` — hàm này tương ứng với cấu hình thứ 4 của chiếc cân của Amina.
  - `A`, `B`, `C`, `D`: Các đồng xu được đặt theo đúng thứ tự trên các đĩa A, B, C, và D tương ứng. `A`, `B`, `C`, và `D` là bốn số nguyên phân biệt, mỗi số có giá trị trong khoảng từ 1 đến 6 bao gồm cả hai giá trị đầu mút.
  - Hàm này trả về một trong các số `A`, `B`, và `C`: số thứ tự của đồng xu mà chiếc cân chọn như được mô tả ở trên theo cấu hình thứ 4. Nghĩa là, đồng xu được trả về là đồng xu có trọng lượng nhẹ nhất trong số các đồng xu trên các đĩa A, B, và C có trọng lượng nặng hơn đồng xu trên đĩa D; hoặc, nếu không có đồng xu nào có trọng lượng nặng hơn đồng xu trên đĩa D, thì đồng xu được trả về đơn giản là đồng xu có trọng lượng nhẹ nhất trong số ba đồng xu trên các đĩa A, B, và C.

Tính điểm

Bài này không có subtask. Thay vào đó, điểm của bạn sẽ phụ thuộc vào bao nhiêu lần cân chương trình của bạn sử dụng (tổng số lần gọi đến các hàm của chương trình chấm: `getLightest()`, `getHeaviest()`, `getMedian()` và/hoặc `getNextLightest()`).

Chương trình của bạn sẽ được chạy nhiều lần, mỗi lần với nhiều test. Gọi r là số lần chạy chương trình của bạn. Con số này được cố định bởi dữ liệu test. Nếu chương trình của bạn không sắp xếp đúng các

đồng xu cho bất kỳ test của bất kỳ lần chạy nào, chương trình sẽ nhận 0 điểm. Ngược lại, điểm của mỗi lần chạy sẽ được tính như sau.

Gọi Q là giá trị nhỏ nhất mà có thể sắp xếp một dãy sáu đồng xu bất kỳ nào sử dụng Q lần cân trên chiếc cân của Amina. Để nhiệm vụ có tính thử thách cao hơn, chúng tôi không cho bạn biết giá trị của Q.

Giả sử số lần cân lớn nhất trong tất cả các test của tất cả các lần chạy là $Q + y$ với giá trị nguyên y nào đó. Khi đó, xét một lần chạy của chương trình của bạn. Gọi số lần cân lớn nhất trong tất cả T test trong lần chạy này là $Q + x$ với một giá trị nguyên không âm x nào đó (Nếu bạn sử dụng ít hơn Q lần cân cho tất cả các test, thì $x = 0$.) Khi đó, điểm của lần chạy này sẽ là $\frac{100}{r((x+y)/5+1)}$, làm tròn *xuống* đến hai chữ số thập phân sau dấu phẩy.

Cụ thể, nếu chương trình của bạn sử dụng nhiều nhất Q lần cân cho mỗi test của mỗi lần chạy, bạn sẽ được 100 điểm.

Ví dụ

Giả sử các đồng xu được sắp xếp theo thứ tự **3 4 6 2 1 5** từ trọng lượng nhẹ nhất đến nặng nhất.

Hàm gọi	Trả về	Giải thích
getMedian(4, 5, 6)	6	Đồng xu **6** có trọng lượng ở giữa trong các đồng xu **4**, **5**, và **6**.
getHeaviest(3, 1, 2)	1	Đồng xu **1** có trọng lượng nặng nhất trong các đồng xu **1**, **2**, and **3**.
getNextLightest(2, 3, 4, 5)	3	Các đồng xu **2**, **3**, và **4** đều có trọng lượng nhẹ hơn đồng xu **5**, nên đồng xu có trọng lượng nhẹ nhất trong chúng (**3**) được trả về.
getNextLightest(1, 6, 3, 4)	6	Các đồng xu **1** và **6** đều có trọng lượng nặng hơn đồng xu **4**. Trong các đồng xu **1** và **6**, đồng xu **6** là đồng xu có trọng lượng nhẹ nhất.
getHeaviest(3, 5, 6)	5	Đồng xu **5** có trọng lượng nặng nhất trong các đồng xu **3**, **5** và **6**.
getMedian(1, 5, 6)	1	Đồng xu **1** có trọng lượng ở giữa trong các đồng xu **1**, **5** và **6**.
getMedian(2, 4, 6)	6	Đồng xu **6** có trọng lượng ở giữa trong các đồng xu **2**, **4** và **6**.
answer([3, 4, 6, 2, 1, 5])		Chương trình tìm ra câu trả lời đúng cho test này.

Chương trình chấm mẫu

Chương trình chấm mẫu đọc dữ liệu vào theo định dạng sau:

- dòng **1**: T —- số lượng các test
- mỗi dòng từ các dòng thứ **2** đến $T + 1$: là một dãy gồm **6** số khác nhau từ **1** đến **6**: thứ tự các đồng xu từ nhẹ nhất đến nặng nhất.

Ví dụ, dữ liệu đầu vào bao gồm hai test mà các đồng xu có thứ tự **1 2 3 4 5 6** và **3 4 6 2 1 5** sẽ được biểu diễn như sau:

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

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

Chương trình chấm mẫu in ra mảng mà được đưa vào là tham số của hàm `answer()`.