ترازو

«امینه» شش سکه دارد که از ۱ تا ۶ شماره‌گذاری شده‌اند. او می‌داند که وزن تمامی سکه‌ها متفاوت است. او دوست دارد که سکه‌ها را برحسب وزن‌شان مرتب کند. به همین منظور، او یک ترازوی جدید طراحی کرده است.

ترازوهای سنتی دارای دو کفه‌اند. برای استفاده از این ترازوها، روی هر کفه یک سکه قرار می‌دهیم و ترازو سکه‌ی سنگین‌تر را مشخص می‌کند.

ترازوی جدید امینه پیچیده‌تر است. این ترازو چهار کفه دارد که با برچسب‌های A، B، C و D مشخص شده‌اند. ترازو چهار «تنظیم» متفاوت دارد، و هر کدام از این تنظیم‌ها، سؤال متفاوتی را در مورد سکه‌ها پاسخ می‌دهد. برای استفاده از ترازو، امینه باید روی هر یک از کفه‌های A، B و C دقیقاً یک سکه قرار دهد. علاوه بر این، در تنظیم چهارم، او باید روی کفه‌ی D هم دقیقاً یک سکه قرار دهد.

ترازو بر اساس چهار تنظیم مذکور به یکی از چهار سؤال زیر پاسخ می‌دهد:

۱. سنگین‌ترین سکه از بین سکه‌های کفه‌های A، B و C کدام است؟

۲. سبک‌ترین سکه از بین سکه‌های کفه‌های A، B و C کدام است؟

۳. کدام‌یک از سکه‌های کفه‌های A، B و C دارای وزن میانه است؟ (منظور سکه‌ای است که نه سنگین‌ترین سکه است و نه سبک‌ترین سکه.)

۴. از میان سکه‌هایی که در کفه‌های A، B و C قرار دارند، تنها سکه‌هایی را در نظر بگیرید که از سکه‌ی کفه‌ی D سنگین‌ترند. اگر چنین سکه‌هایی وجود داشته باشند، سبک‌ترین سکه از بین این سکه‌ها کدام است؟ اگر چنین سکه‌هایی وجود نداشته باشند، سبک‌ترین سکه از بین سکه‌های کفه‌های A، B و C کدام است؟

مسئله

برنامه‌ای بنویسید که سکه‌های امینه را بر حسب وزن‌شان مرتب کند. برنامه‌ی شما می‌تواند برای مقایسه‌ی سکه‌ها از ترازوی امینه استفاده کند. به برنامه‌ی شما چند «مورد آزمون»[1] داده خواهد شد که هر یک، متناظر با مجموعه‌ای جدید از شش سکه است.

برنامه‌ی شما باید توابع `init` و `orderCoins` را پیاده‌سازی کند. در هر اجرای برنامه‌ی شما، ارزیاب ابتدا تابع `init` را دقیقاً یک بار فراخوانی می‌کند. این فراخوانی تعداد موارد آزمون را به شما می‌دهد و به‌کمک آن می‌توانید متغیرهایتان را مقداردهی اولیه کنید. سپس، ارزیاب به‌ازای هر مورد آزمون، یک‌بار تابع `orderCoins()` را فراخوانی می‌کند.

[1] Test case

- `init(T)`
  - `T`: تعداد موارد آزمونی است که برنامه‌ی شما باید در این اجرا پاسخ دهد. `T` یک عدد صحیح در محدوده‌ی ۱ تا ۱۸ است.
  - این تابع مقداری به عنوان خروجی برنمی‌گرداند.
- `orderCoins()`
  - این تابع به ازای هر مورد آزمون، دقیقاً یک‌بار فراخوانی می‌شود.
  - این تابع باید ترتیب درست سکه‌های امینه را با استفاده از فراخوانی توابع ارزیاب `getHeaviest()`، `getLightest()`، `getMedian()` و `getNextLightest()` مشخص کند.
  - زمانی که تابع، ترتیب درست سکه‌ها را به‌دست می‌آورد، باید آن را با فراخوانی تابع ارزیاب `answer()` گزارش دهد.
  - پس از فراخوانی `answer()`، تابع `orderCoins()` باید خاتمه یابد. این تابع مقداری برنمی‌گرداند.

شما می‌توانید از توابع ارزیاب زیر استفاده کنید:

- `answer(W)` — برنامه‌ی شما باید از این تابع برای گزارش جوابی که پیدا کرده است استفاده کند.
  - `W`: آرایه‌ای به طول ۶، شامل ترتیب درست سکه‌ها. `W[0]` تا `W[5]` باید شماره‌ی سکه‌ها (یعنی اعداد ۱ تا ۶) به ترتیب از سبک‌ترین به سنگین‌ترین سکه باشد.
  - برنامه‌ی شما باید این تابع را فقط از داخل `orderCoins()` یک بار به‌ازای هر مورد آزمون فراخوانی کند.
  - این تابع مقداری برنمی‌گرداند.
- `getHeaviest(A, B, C)`، `getLightest(A, B, C)`، `getMedian(A, B, C)` — این توابع به‌ترتیب متناظر با تنظیم‌های شماره‌ی ۱، ۲ و ۳ برای ترازوی امینه هستند.
  - `A`، `B`، `C`: سکه‌هایی که به ترتیب روی کفه‌های A، B و C قرار می‌گیرند. `A`، `B` و `C` باید سه عدد صحیح متمایز در محدوده‌ی ۱ تا ۶ باشند.
  - هر یک از توابع یکی از اعداد `A`، `B` و `C` را برمی‌گرداند که شماره‌ی سکه‌ی خواسته‌شده است. برای مثال، تابع `getHeaviest(A, B, C)` شماره‌ی سنگین‌ترین سکه از بین سه سکه‌ی داده شده را می‌دهد.
- `getNextLightest(A, B, C, D)` — این تابع متناظر با تنظیم شماره‌ی ۴ برای ترازوی امینه است.
  - `A`، `B`، `C`، `D`: سکه‌هایی که به ترتیب بر روی کفه‌های A، B، C و D قرار گرفته‌اند. `A`، `B`، `C`،و `D` باید چهار عدد صحیح متمایز در محدوده‌ی ۱ تا ۶ باشند.
  - این تابع یکی از اعداد `A`، `B` و `C` را برمی‌گرداند که شماره‌ی سکه‌ای است که توسط ترازو در تنظیم شماره‌ی ۴ مطابق توضیح فوق انتخاب می‌شود. بدین معنی که سکه‌ی برگردانده‌شده سبک‌ترین سکه از بین سکه‌های کفه‌های A، B و C است که از سکه‌ی کفه‌ی D سنگین‌تر هستند؛ و یا در صورتی که هیچ‌یک از این سکه‌ها از سکه‌ی کفه‌ی D سنگین‌تر نباشند، سکه‌ی برگردانده‌شده سبک‌ترین سکه از بین تمامی سکه‌های کفه‌های A، B و C است.

امتیازدهی

این سؤال هیچ زیرمسئله‌ای ندارد. در عوض، امتیاز شما برحسب تعداد توزین‌های برنامه (یعنی تعداد فراخوانی‌های توابع ارزیاب `getLightest()`، `getHeaviest()`، `getMedian()`، و `getNextLightest()`) مشخص می‌شود.

برنامه‌ی شما چند بار اجرا می‌شود و در هر اجرا، چند مورد آزمون به آن داده می‌شود. فرض کنید r تعداد اجراهای برنامه باشد. اگر برنامه‌ی شما حتی در یکی از موارد آزمونِ یکی از اجراها ترتیب درست را برنگرداند، امتیاز صفر به برنامه داده می‌شود. در غیر این‌صورت، امتیاز اجراها به طور جداگانه به صورت زیر محاسبه می‌شود.

فرض کنید Q کوچک‌ترین عددی است که مرتب‌کردن هر دنباله‌ای از شش سکه، با استفاده از Q بار توزین با ترازوی امینه ممکن باشد. برای این که مسئله چالشی‌تر شود، مقدار Q را در این‌جا مشخص نمی‌کنیم.

فرض کنید بیش‌ترین تعداد توزین‌ها در میان تمامی موارد آزمونِ تمامی اجراها برابر با $Q + y$ (به ازای یک عدد صحیح y) باشد. حال، یک اجرا از برنامه‌ی خود را در نظر بگیرید. فرض کنید بیش‌ترین تعداد توزین در بین T مورد آزمون این اجرا برابر $Q + x$ (به ازای یک عدد صحیح نامنفی x) باشد. (در صورتی که تعداد دفعات توزین برنامه‌ی شما برای تمام موارد آزمون کم‌تر از Q باشد، آن‌گاه $x = ۰$ است.) در این صورت، امتیاز شما برای این اجرا برابر $\frac{۱۰۰}{r(\frac{x+y}{۵}+۱)}$ است که این عدد تا دو رقم اعشار به پایین گرد می‌شود.

به طور خاص، اگر برنامه‌ی شما به ازای هر یک از موارد آزمونِ تمامی اجراها، حداکثر از Q مرتبه توزین استفاده کند، نمره‌ی شما ۱۰۰ می‌شود.

مثال

فرض کنید سکه‌ها به ترتیب (از چپ به راست) ۵ ۱ ۲ ۶ ۴ ۳ از سبک‌ترین تا سنگین‌ترین باشند.

فراخوانی تابع	خروجی	توضیح
`getMedian(4,5,6)`	6	سکه‌ی ۶ میانه‌ی سکه‌های ۴، ۵ و ۶ است.
`getHeaviest(3,1,2)`	1	سکه‌ی ۱ سنگین‌ترین سکه میان سکه‌های ۱، ۲ و ۳ است.
`getNextLightest(2,3,4,5)`	3	از میان سکه‌های ۲، ۳ و ۴ سبک‌ترین سکه‌ی سنگین‌تر از ۵، سکه‌ی ۳ است.
`getNextLightest(1,6,3,4)`	6	از میان سکه‌های ۱، ۶ و ۳ سبک‌ترین سکه‌ی سنگین‌تر از ۴، سکه‌ی ۶ است.
`getHeaviest(3,5,6)`	5	سکه‌ی ۵ سنگین‌ترین سکه میان سکه‌های ۳، ۵ و ۶ است.
`getMedian(1,5,6)`	1	سکه‌ی ۱ میانه‌ی سکه‌های ۱، ۵ و ۶ است.
`getMedian(2,4,6)`	6	سکه‌ی ۶ میانه‌ی سکه‌های ۲، ۴ و ۶ است.
`answer([3,4,6,2,1,5])`		برنامه پاسخ درست را برای این مورد آزمون پیدا کرده است.

ارزیاب نمونه

ارزیاب نمونه ورودی را با فرمت زیر می‌خواند:

- سطر ۱: مقدار T — تعداد موارد آزمون

- هر یک از سطرهای ۲ تا $T+1$: دنباله‌ای از ۶ عدد طبیعی متمایز در محدوده‌ی ۱ تا ۶ — ترتیب سکه‌ها از سبک‌ترین به سنگین‌ترین

برای مثال، ورودی زیر شامل دو مورد آزمون با ترتیب‌های ۱ ۲ ۳ ۴ ۵ ۶ و ۳ ۴ ۶ ۲ ۱ ۵ است:

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

ارزیاب نمونه آرایه‌ای را که به عنوان پارامتر به تابع `answer()` داده می‌شود، چاپ می‌کند.