Terazi

Amina'nın 1'den 6'ya numaralandırılmış altı tane bozuk parası vardır. Amina her bir bozuk parasının farklı ağırlığa sahip olduğunu bilmektedir. Paralarını ağırlıklarına göre sıralamak istemektedir. Bu amaç için yeni bir tip terazi geliştirmiştir.

Klasik bir terazinin iki kefesi vardır. Klasik bir teraziyi kullanmak için, her bir kefeye birer bozuk para yerleştirirsiniz ve terazi size hangi paranın daha ağır olduğunu gösterir.

Amina'nın yeni terazisi daha karmaşıktır. A, B, C, ve D şeklinde etiketlenmiş dört tane kefesi vardır. Terazinin ayrıca dört farklı ayarı vardır ve her ayar farklı bir soruyu yanıtlamak için kullanılır. Teraziyi kullanmak için Amina A, B, ve C kefelerine tam olarak birer tane para koymalıdır. Ayrıca, dördüncü ayarda, D kefesine de tam olarak bir tane para koymalıdır.

Terazinin dört ayarı sırasıyla aşağıdaki dört soruya cevap bulmak için kullanılır:

1. A, B, ve C kefelerinden hangi kefedeki para en ağırdır?
2. A, B, ve C kefelerinden hangi kefedeki para en hafiftir?
3. A, B, ve C kefelerinden hangi kefedeki para orta ağırlıktadır? (Bu para üç para içinde en ağır ya da en hafif olmayan paradır.)
4. A, B, ve C kefelerindeki paralar arasından, D kefesindeki paradan ağır olanlarını göz önüne alalım. Bu paralar arasında en hafifi hangi kefedeki paradır? Eğer D kefesindeki paradan ağır para yoksa, A, B, ve C kefelerindeki paralardan hangisi en hafiftir?

Görev

Amina'nın altı bozuk parasını ağırlıklarına göre sıralamak için bir program yazınız. Programınız Amina'nın terazisine sorgular yönelterek paraların ağırlıklarını kıyaslayacaktır. Programınıza birçok test girdisi verilecektir ve her bir girdi farklı bir bozuk para kümesini temsil edecektir.

Programınız `init` ve `orderCoins` fonksiyonlarını gerçekleştirmelidir. Programınızın her çalışmasında, grader ilk olarak `init` fonksiyonunu yalnızca bir kez çağıracaktır. Bu çağrı sonucu kaç tane test girdisi olduğunu bilecek ve değişkenlerinize ilk değerlerini atayabileceksiniz. Grader daha sonra her bir test girdisi için `orderCoins()` fonksiyonunu çağıracaktır.

- `init(T)`
  - `T`: Programınızın bu çalıştırmada çözmesi gereken test girdisi sayısı. `T`, $1, \ldots, 18$ arası bir tamsayıdır.
  - Bu fonksiyon herhangi bir değer döndürmez.
- `orderCoins()`
  - Bu fonksiyon her bir test girdisi için tam olarak bir kez çağrılır.

- Bu fonksiyon Amina'nın paralarının doğru sırasını `getHeaviest()`, `getLightest()`, `getMedian()`, ve/veya `getNextLightest()` grader fonksiyonlarını çağırarak belirler.
- Fonksiyon doğru sırayı bulduğu zaman, bu sırayı `answer()` adlı grader fonksiyonunu çağırarak bildirmelidir.
- `answer()` fonksiyonunu çağırdıktan sonra, `orderCoins()` fonksiyonu sonlanmalıdır. `orderCoins()` fonksiyonu bir değer dönmez.

Programınızda aşağıdaki grader fonksiyonlarını kullanabilirsiniz:

- `answer(W)` — programınız bu fonksiyonu kullanarak bulduğu cevabı bildirmelidir.
  - `W`: 6 uzunluğunda bir dizidir ve paraların doğru sırasını gösterir. Dizi içeriği `W[0]`'dan `W[5]`'e en hafiften en ağıra doğru paraların numaralarını gösterir (para numaraları 1'le 6 arasındadır).
  - Programınız bu fonksiyonu `orderCoins()` fonksiyonu içerisinden her bir test girdisi için sadece bir kez çağırmalıdır.
  - Bu fonksiyon bir değer dönmez.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — bu fonksiyonlar Amina'nın terazisindeki 1, 2 ve 3 ayarlarına karşılık gelir.
  - `A`, `B`, `C`: Sırasıyla, A, B, ve C kefelerine konan paraların numaralarını gösterir. `A`, `B`, ve `C` herbiri 1 ve 6 arasında (sınırlar dahil) birbirinden farklı üç tamsayı olmalıdır.
  - Her bir fonksiyon `A`, `B`, ve `C` sayılarından birisini döndürür: dönen sayı cevap olan paranın numarasıdır. Mesela, `getHeaviest(A, B, C)` çağrısı verilen üç para içerisinde en ağır olanın numarasını döner.
- `getNextLightest(A, B, C, D)` — bu fonksiyon terazide 4. ayara karşılık gelir.
  - `A`, `B`, `C`, `D`: Sırasıyla, A, B, C, ve D kefelerine konan paraların numaralarıdır. `A`, `B`, `C`, ve `D` herbiri 1 ve 6 arasında (sınırlar dahil) birbirinden farklı dört tamsayı olmalıdır.
  - Bu fonksiyon `A`, `B`, ve `C` sayılarından birisini döndürür: yukarıda açıklanan 4. ayarın cevabı olan paranın numarası döndürülür. Yani, dönen para A, B, ve C içinde D'deki kefedeki paradan ağır olanlardan en hafifidir; ya da, D'deki kefedeki paradan ağır para yok ise, basitçe A, B, ve C kefelerindeki paraların en hafifidir.

Notlandırma

Bu soruda altgörev bulunmamaktadır. Sorudan alacağınız puan yaptığınız ağırlık sorgularının toplam sayısına göre belirlenecektir (yani `getLightest()`, `getHeaviest()`, `getMedian()` ve/veya `getNextLightest()` grader fonksiyonlarına yaptığınız çağrıların toplamına göre).

Programınız birçok kez ve her seferinde birçok test girdisi ile çalıştırılacaktır. Programınız r kez çalıştırılmış olsun. Test verisine göre bu sayı sabit bir sayı olacaktır. Eğer programınız herhangi bir çalışmada herhangi bir test girdisini yanlış sıralarsa 0 puan alacaktır. Bütün test girdilerini doğru sıraladığı takdirde, programın her bir çalışması ayrı ayrı aşağıdaki gibi puanlandırılacaktır.

Q, herhangi altı tane bozuk parayı Amina'nın terazisini kullanarak sıralamak için gereken en küçük

sayıda ağırlık sorgusu olsun. Problemi zorlaştırmak için Q'nun kaç olduğunu burada açıklamıyoruz.

Programınızın bütün çalışmalarındaki bütün test girdileri için kullanılan en fazla sorgu sayısı $Q + y$ olsun (y herhangi bir tamsayı). Programınızın tek bir çalışmasını göz önüne alalım. Bu çalışmadaki T tane test girdisinden en fazla ağırlık sorgusu kullanılan girdi $Q + x$ tane sorgu kullanmış olsun (x negatif olmayan bir tamsayı). (Eğer her bir test girdisi için Q'dan az sayıda sorgu kullanmışsanız o zaman $x = 0$'dır.) Bu durumda bu çalışmadan alacağınız puan $\frac{100}{r((x+y)/5+1)}$ olacaktır ve iki basamak aşağıya yuvarlanacaktır.

Yani, eğer programınız her bir çalışmadaki her bir test girdisi için en fazla Q adet ağırlık sorgusu yaparsa, bu problemden 100 puan alırsınız.

Örnek

Bozuk paraların en hafiften en ağıra $3\ 4\ 6\ 2\ 1\ 5$ şeklinde sıralandığını düşünelim.

Çağrılan fonksiyon	Dönen değer	Açıklama
getMedian(4, 5, 6)	6	**6** nolu para **4**, **5**, ve **6** arasında ortada olandır.
getHeaviest(3, 1, 2)	1	**1** nolu para **1**, **2**, ve **3** arasında en ağırıdır.
getNextLightest(2, 3, 4, 5)	3	**2**, **3**, ve **4** nolu paraların hepsi **5** nolu paradan hafiftir, bu yüzden içlerindeki en hafif olanı (**3**) sonuç olarak döner.
getNextLightest(1, 6, 3, 4)	6	**1** ve **6** nolu paralar **4** nolu paradan ağırdır. **1** ve **6**'dan **6** nolu para hafif olandır.
getHeaviest(3, 5, 6)	5	**5** nolu para **3**, **5** ve **6** arasında en ağırıdır.
getMedian(1, 5, 6)	1	**1** nolu para **1**, **5** ve **6** arasında ortada olandır.
getMedian(2, 4, 6)	6	**6** nolu para **2**, **4** ve **6** arasında ortada olandır.
answer([3, 4, 6, 2, 1, 5])		Program bu test girdisi için cevabı doğru bulmuştur.

Örnek grader

Örnek grader girdiyi aşağıdaki formatta okur:

- satır 1: T —- test girdisi sayısı
- 2'den $T + 1$'e olan her bir satır: 1'den 6'ya birbirinden farklı 6 sayı içerir: paraların en hafiften en ağıra doğru sıralamalarını gösterir.

Mesela, paraların $1\ 2\ 3\ 4\ 5\ 6$ ve $3\ 4\ 6\ 2\ 1\ 5$ şeklinde sıralandığı iki test girdisi içeren bir girdi şu şekilde gözükür:

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

Örnek grader `answer()` fonksiyonuna parametre olarak verilen diziyi ekrana basar.