Balances

Amina possède six pièces, numérotées de 1 à 6. Elle sait que toutes les pièces ont des poids différents. Elle aimerait pouvoir les trier selon leur poids. À cette fin, elle a développé un nouveau type de balance.

Une balance traditionelle a deux plateaux. Afin de l'utiliser, il faut placer une pièce sur chaque plateau et la balance indiquera laquelle est la plus lourde.

La nouvelle balance d'Amina est plus complexe. Elle a quatre plateaux, dénommés A, B, C et D. La balance a quatre configurations différentes, chacune répondant à une question différente à propos des pièces. Afin d'utiliser la balance, Amina doit placer exactement une pièce sur chacun des plateaux A, B et C. De plus, la quatrième configuration nécessite de placer exactement une pièce sur le plateau D.

Les quatre configurations permettent à la balance de répondre aux quatre questions suivantes :

1. Laquelle des pièces des plateaux A, B et C est la plus lourde ?
2. Laquelle des pièces des plateaux A, B et C est la plus légère ?
3. Laquelle des pièces des plateaux A, B et C est la médiane ? (la pièce qui n'est ni la plus lourde, ni la plus légère des trois)
4. Parmi les pièces des plateaux A, B et C, ne considérer que les pièces plus lourdes que celle du plateau D. Si il y en au moins une, laquelle est la plus légère ? Sinon, quelle est la pièce la plus légère parmi les pièces des plateaux A, B et C ?

Tâche

Écrivez un programme qui ordonne les six pièces d'Amina selon leur poids. Le programme peut interroger la balance d'Amina pour comparer le poids des pièces. Votre programme devra résoudre plusieurs scénarios, chacun correspondant à un différent ensemble de six pièces.

Votre programme doit implémenter les fonctions `init` et `orderCoins`. Pour chaque exécution de votre programme, l'évaluateur appellera `init` une seule fois. Cela vous fournira le nombre de scénarios et vous permettra d'initialiser vos variables. L'évaluateur appellera ensuite `orderCoins()` une fois par scénario.

- `init(T)`
  - `T` : Le nombre de scénarios que votre programme devra résoudre lors de cette exécution. `T` est un entier dans l'intervalle $1, \ldots, 18$.
  - Cette fonction ne doit rien retourner.
- `orderCoins()`

- Cette fonction est appelée une seule fois par scénario.
- La fonction doit déterminer l'ordre des pièces d'Amina à l'aide des fonctions `getHeaviest()`, `getLightest()`, `getMedian()` et `getNextLightest()`.
- Une fois que la fonction connaît l'ordre des pièces, elle doit appeler la fonction `answer()`.
- Après avoir appelé `answer()`, la fonction `orderCoins()` doit s'arrêter. Elle ne doit rien retourner.

Vous pouvez utiliser les fonctions suivantes de l'évaluateur (grader) dans votre programme :

- `answer(W)` — votre programme doit utiliser cette fonction pour rapporter la réponse trouvée.
  - `W` : Un tableau de taille 6 contenant l'ordre correct des pièces. `W[0]` à `W[5]` doivent être des numéros de pièces (c-à-d. des nombres de 1 à 6) de la plus légère à la plus lourde.
  - Votre programme ne doit appeler cette fonction, depuis `orderCoins()`, qu'une seule fois par scénario.
  - Cette fonction ne retourne rien.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — qui correspondent aux configurations 1, 2 et 3 de la balance d'Amina.
  - `A`, `B`, `C` : Les pièces posées respectivement sur les plateaux A, B et C. `A`, `B` et `C` doivent être trois entiers distincts, chacun compris entre 1 et 6 inclus.
  - Chaque fonction retourne l'un des nombres `A`, `B` ou `C` : le numéro de la pièce appropriée. Par exemple, `getHeaviest(A, B, C)` retourne le numéro de la pièce la plus lourde parmi les trois pièces fournies.
- `getNextLightest(A, B, C, D)` — qui correspond à la configuration 4 de la balance d'Amina.
  - `A`, `B`, `C`, `D` : Les pièces posées respectivement dans les plateaux A, B, C et D. `A`, `B`, `C` et `D` doivent être quatre entiers distincts, chacun compris entre 1 et 6 inclus.
  - La fonction retourne l'un des nombres `A`, `B` ou `C` : le numéro de pièce déterminé par la balance dans la configuration 4 décrite plus tôt, c'est-à-dire la plus légère des pièces des plateaux A, B et C qui sont plus lourdes que la pièce en D, ou, si aucune des pièces n'est plus lourde que D, la plus légère des trois plateaux A, B et C.

Scores

Il n'y a pas de sous-tâche dans ce problème. Cependant, votre score sera basé sur le nombre de pesées (nombre d'appels à `getLightest()`, `getHeaviest()`, `getMedian()` et `getNextLightest()`) que votre programme effectuera.

Votre programme sera exécuté plusieurs fois, chaque fois sur plusieurs scénarios. Soit r le nombre d'exécutions de votre programme. Ce nombre est défini dans les données de test. Si votre programme ordonne incorrectement l'un des scénarios de l'une des exécutions, vous obtiendrez 0 point. Sinon, chaque exécution est évaluée comme suit.

Soit Q le nombre le plus petit tel qu'il est possible de trier n'importe quelle séquence en faisant Q pesées sur la balance d'Amina. Afin de rendre la tâche plus difficile, on ne révèle pas la valeur de Q.

Supposons que le plus grand nombre de pesées parmi tous les scénarios de toutes les exécutions est $Q + y$ pour un entier y. Chaque exécution de votre programme est alors évaluée comme suit. Si $Q + x$ est le plus grand nombre de pesées parmi les T scénarios de cette exécution, avec $x >= 0$ (si vous utilisez moins de Q pesées pour chaque scénario, alors $x = 0$), alors le score de cette exécution sera $\frac{100}{r((x+y)/5+1)}$, arrondi vers le bas à deux décimales.

En particulier, si votre programme fait au plus Q pesées pour chaque scénario de chaque exécution, vous obtiendrez 100 points.

Exemple

Considérons que les pièces sont ordonnées **3 4 6 2 1 5** de la plus légère à la plus lourde.

Appel de fonction	Retour	Explication
getMedian(4, 5, 6)	6	La pièce **6** est la médiane parmi les pièces **4**, **5** et **6**.
getHeaviest(3, 1, 2)	1	La pièce **1** est la plus lourde parmi les pièces **1**, **2** et **3**.
getNextLightest(2, 3, 4, 5)	3	Les pièces **2**, **3** et **4** sont toutes plus légères que **5**, la plus légère des trois (**3**) est donc retournée.
getNextLightest(1, 6, 3, 4)	6	Les pièces **1** et **6** sont toutes deux plus lourdes que la pièce **4**. Parmi les pièces **1** et **6**, la pièce **6** est la plus légère.
getHeaviest(3, 5, 6)	5	La pièce **5** est la plus lourde des pièces **3**, **5** et **6**.
getMedian(1, 5, 6)	1	La pièce **1** est la médiane parmi les pièces **1**, **5** et **6**.
getMedian(2, 4, 6)	6	La pièce **6** est la médiane parmi les pièces **2**, **4** et **6**.
answer([3, 4, 6, 2, 1, 5])		Le programme a trouvé la réponse correcte pour ce scénario.

Évaluateur fourni (grader)

L'évaluateur fourni lit son entrée dans le format suivant :

- ligne **1** : T —- le nombre de scénarios
- chaque ligne de **2** à $T + 1$: une séquence de **6** nombres distincts de **1** à **6** : l'ordre des pièces de la plus légère à la plus lourde.

Par exemple, pour les deux scénarios où les pièces sont ordonnées **1 2 3 4 5 6** et **3 4 6 2 1 5**, l'entrée doit être :

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

L'évaluateur fourni écrit le tableau passé en paramètre à la fonction `answer()`.