Scales

Amina has seks mynter, nummerert fra 1 til 6. Hun vet at ingen av myntene veier det samme. Hun ønsker å ordne myntene etter vekt. For å gjøre dette har hun utviklet en ny type balansevekt.

En tradisjonell balansevekt har to skåler. For å bruke en slik vekt så legger man én mynt på hver skål og vekten vil fortelle deg hvilken mynt som er tyngst.

Aminas nye vekt er mer avansert. Den har fire skåler, merket A, B, C, og D. Vekten har fire forskjellige instillinger som kan brukes til å stille forskjellige spørsmål om myntene. For å bruke vekten må Amina legge nøyaktig én mynt i hver av skålene A, B og C. Dersom hun vil bruke den fjerde instillingen, så må hun også legge nøyaktig én mynt i skål D.

De fire instillingene vil instruere vekten til å svare på de følgende fire spørsmålene:

1. Hvilken av myntene i skålene A, B, og C er den tyngste?
2. Hvilken av myntene i skålene A, B, og C er den letteste?
3. Hvilken av myntene i skålene A, B, og C er medianen? (Dette er den mynten som er hverken den tyngste eller letteste blandt de tre.)
4. Av myntene i skålene A, B og C, betrakt kun de myntene som er tyngre enn mynten i skål D. Dersom det finnes noen slike mynter, hvilken av disse myntene er den letteste? Dersom det ikke er noen slike mynter, hvilken av myntene i skålene A, B, og C er den letteste?

Task

Skriv et program som sorterer Aminas seks mynter etter vekt. Programmet ditt kan spørre Aminas vekt for å sammenligne hvor mye myntene veier. Programmet ditt vil bli gitt flere testsett som skal løses. Hver av disse testsettene tilsvarer et nytt sett med seks mynter.

Programmet ditt skal implementere funksjonene `init` og `orderCoins`. For hver kjøring av programmet ditt vil graderen først kalle `init` nøyaktig én gang. Dette gir deg antall testsett i denne kjøringen og lar deg initialisere eventuelle variabler du måtte ønske. Deretter vil graderen kalle `orderCoins()` én gang per testsett.

- `init(T)`
  - `T`: Antall testsett som programmet ditt må løse i denne kjøringen. `T` er et heltall i intervallet $1, \ldots, 18$.
  - Denne funksjonen har ingen returverdi
- `orderCoins()`
  - Denne funksjonen kalles nøyaktig én gang per testsett.
  - Denne funksjonen skal avgjøre riktig rekkefølge av myntene ved å kalle funksjonene

`getHeaviest()`, `getLightest()`, `getMedian()`, og/eller `getNextLightest()`, som alle vil være implementert av graderen.

- Når funksjonen har klart å finne den riktige rekkefølgen så skal den rapportere denne til graderen ved å kalle funksjonen `answer()`
- Etter at den har kallt `answer()`, så skal funksjonen `orderCoins()` returnere. Den har ingen returverdi.

Du kan bruke disse grader funksjonene fra programmet ditt:

- `answer(W)` — Programmet ditt skal bruke denne til å rapportere svaret på et testsett
  - `W`: En array med lengde 6 inneholdende myntene i korrekt rekkefølge. `W[0]` til `W[5]` skal være nummerene på myntene (dvs. tallene fra 1 til 6) i rekkefølge fra den letteste mynten til den tyngste mynten.
  - Programmet ditt skal kun kalle denne funksjonen fra `orderCoins()`, og kun én gang per testsett.
  - Denne funksjonen har ingen returverdi.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — Disse tilsvarer henholdsvis instillingene 1, 2 og 3 på Amina's vekt.
  - `A`, `B`, `C`: Myntene som legges i skålene A, B, og C, henholdsvis. `A`, `B`, og `C` skal være tre forskjellige heltall, hver av de fra intervallet $1, \ldots, 6$.
  - Hver funksjon returnerer ett av tallene `A`, `B`, eller `C`: tallet til den mynten som svarer på spørringen. For eksempel så vil, `getHeaviest(A, B, C)` returnere tallet på den tyngste av de tre gitte myntene.
- `getNextLightest(A, B, C, D)` — Denne tilsvarer instilling 4 for Amina's vekt.
  - `A`, `B`, `C`, `D`: Myntene som legges i skålene A, B, C, og D, henholdsvis. `A`, `B`, `C`, og `D` skal være fire forskjellige heltall, hver av de fra intervallet $1, \ldots, 6$.
  - Funksjonen returnerer ett av tallene `A`, `B`, og `C`: tallet på mynten valgt av vekten som beskrevet ovenfor for instilling 4. Dvs. den returnerte mynten er den letteste av de myntene i skålene A, B, og C som er tyngre enn mynten i skål D; eller - dersom ingen av de er tyngre enn mynten i skål D, så vil den returnerte mynten være den letteste av de tre myntene i skålene A, B, og C.

Scoring

There are no subtasks in this problem. Instead, your score will be based on how many weighings (total number of calls to grader functions `getLightest()`, `getHeaviest()`, `getMedian()` and/or `getNextLightest()`) your program makes.

Your program will be run multiple times with multiple test cases in each run. Let r be the number of runs of your program. This number is fixed by the test data. If your program does not order the coins correctly in any test case of any run, it will get 0 points. Otherwise, the runs are scored individually as follows.

Let Q be the smallest number such that it is possible to sort any sequence of six coins using Q

weighings on Amina's scale. To make the task more challenging, we do not reveal the value of Q here.

Suppose the largest number of weighings amongst all test cases of all runs is $Q + y$ for some integer y. Then, consider a single run of your program. Let the largest number of weighings amongst all T test cases in this run be $Q + x$ for some non-negative integer x. (If you use fewer than Q weighings for every test case, then $x = 0$.) Then, the score for this run will be $\frac{100}{r((x+y)/5+1)}$, rounded *down* to two digits after the decimal point.

In particular, if your program makes at most Q weighings in each test case of every run, you will get 100 points.

Example

Suppose the coins are ordered **3 4 6 2 1 5** from the lightest to the heaviest.

Function call	Returns	Explanation
getMedian(4, 5, 6)	6	Coin **6** is the median among coins **4**, **5**, and **6**.
getHeaviest(3, 1, 2)	1	Coin **1** is the heaviest among coins **1**, **2**, and **3**.
getNextLightest(2, 3, 4, 5)	3	Coins **2**, **3**, and **4** are all lighter than coin **5**, so the lightest among them (**3**) is returned.
getNextLightest(1, 6, 3, 4)	6	Coins **1** and **6** are both heavier than coin **4**. Among coins **1** and **6**, coin **6** is the lightest one.
getHeaviest(3, 5, 6)	5	Coin **5** is the heaviest among coins **3**, **5** and **6**.
getMedian(1, 5, 6)	1	Coin **1** is the median among coins **1**, **5** and **6**.
getMedian(2, 4, 6)	6	Coin **6** is the median among coins **2**, **4** and **6**.
answer([3, 4, 6, 2, 1, 5])		The program found the right answer for this test case.

Sample grader

The sample grader reads input in the following format:

- line **1**: T —- the number of test cases
- each of the lines from **2** to $T + 1$: a sequence of **6** distinct numbers from **1** to **6**: the order of the coins from the lightest to the heaviest.

For instance, an input that consists of two test cases where the coins are ordered **1 2 3 4 5 6** and **3 4 6 2 1 5** looks as follows:

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

The sample grader prints the array that was passed as a parameter to the `answer()` function.