Vågar

Amina har sex mynt, numrerade från 1 till 6. Hon vet att alla mynten har olika vikt. Hon skulle vilja ordna dem utifrån deras vikt. För att göra detta har hon utvecklat en ny sorts balansvåg.

En vanlig balansvåg har två skålar. Den används genom att man lägger ett mynt i varje vågskål, och vågen kommer då att tala om vilket mynt som är tyngst.

Amina nya våg är mer komplicerad. Den har fyra skålar, märkta A, B, C och D. Vågen har fyra olika inställningar, som alla besvarar olika frågor rörande mynten. För att använda vågen måste Amina placera exakt ett mynt i endera av skålarna A, B och C. För den fjärde inställningen måste hon dessutom lägga precis ett mynt i skål D.

De fyra inställningarna får vågen att besvara motsvarande fråga bland de nedstående:

1. Vilket av mynten i skålarna A, B och C är det tyngsta?
2. Vilket av mynten i skålarna A, B och C är det lättaste?
3. Vilket av mynten i skålarna A, B och C är medianen? (D.v.s., myntet som varken är lättast eller tyngst av de tre.)
4. Bland mynten i skålarna A, B och C, betrakta enbart de mynt som är tyngre än myntet i skål D. Om det finns några, vilket av dem är lättast? Annars, om inget av mynten är tyngre än det i skål D, vilket av mynten i skålarna A, B och C är det lättaste?

Uppgift

Skriv ett program som ordnar Aminas sex mynt enligt deras vikt. Programmet kan använda sig av Aminas våg för att jämföra vikter på mynt. Programmet kommer att ges flera testfall att lösa, vardera svarandes mot en ny samling av sex mynt.

Programmet skall implementera funktionerna `init` och `orderCoins`. Under varje körning av programmet kommer rättaren först att anropa `init` exakt en gång. Det här ger dig antalet testfall och låter dig initialisera de variabler du känner för. Rättaren kommer sedan att anropa `orderCoins()` en gång per testfall.

- `init(T)`
  - `T`: Antal testfall som programmet kommer att behöva lösa under den här körningen. `T` är ett heltal i intervallet $1, \ldots, 18$.
  - Den här funktionen har inget returvärde.
- `orderCoins()`
  - Den här funktionen anropas exakt en gång per testfall.
  - Funktionen skall ta reda på ordningen av Aminas mynt genom anrop till rättarfunktionerna

`getHeaviest()`, `getLightest()`, `getMedian()`, och/eller `getNextLightest()`.

- Efter att funktionen tagit reda på den korrekta ordningen skall den reportera detta genom att anropa rättarfunktionen `answer()`.
- Efter att den anropat `answer()` ska funktionen `orderCoins()` returnera. Den har inget returvärde.

Du får använda följande graderfunktioner i ditt program:

- `answer(W)` — ditt program bör använda den här funktionen för att rapportera svaret den hittat.
  - `W`: En array av längd 6 som innehåller en korrekt ordning av mynten. `W[0]` upp till `W[5]` ska vara myntnumren (d.v.s., tal från 1 till 6) i ordning från den lättaste till det tyngsta myntet.
  - Ditt program ska enbart anropa den här funktionen från `orderCoins()`, en gång per testfall.
  - Den här funktionen har inget returvärde.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — dessa motsvarar inställningar 1, 2 och 3 respektive på Aminas våg.
  - `A`, `B`, `C`: Mynten som läggs i skålar A, B och C, respektive. `A`, `B` och `C` ska vara tre distinkta heltal, vardera mellan 1 och 6 inklusive.
  - Funktionerna returnerar ett av talen `A`, `B` och `C`: siffran på det relevanta myntet. Till exempel returnerar `getHeaviest(A, B, C)` siffran på det tyngsta av de tre givna mynten.
- `getNextLightest(A, B, C, D)` — detta motsvarar inställning 4 på Aminas våg
  - `A`, `B`, `C`, `D`: Mynten som läggs i skålar A, B, C och D, respektive. `A`, `B`, `C` och `D` ska vara fyra distinkta heltal, vardera mellan 1 och 6 inklusive.
  - Funktionen returnerar ett av talen `A`, `B` och `C`: siffran på det mynt som vågen svarar med som beskrivet ovan för inställning 4. D.v.s., det returnerade myntet är det lättaste bland de mynt i skålar A, B och C som är tyngre än myntet i skål D, eller, om inget av dem är tyngre än myntet i skål D, är det returnerade myntet helt enkelt det lättaste av alla mynt i skålar A, B och C.

Poängsättning

Det finns inga deluppgifter för det här problem. I stället bestäms din poäng baserat på hur många vägningar (antal anrop till rättarfunktionerna `getLightest()`, `getHeaviest()`, `getMedian()` och/eller `getNextLightest()`) som ditt program utför.

Ditt program kommer att köras flera gånger, med flera testfall i varje körning. Låt r vara antalet körningar av programmet (det här är fixt och bestäms av testdatat). Om ditt program misslyckas med att ordna mynten korrekt i något testfall för någon körning kommer det att få 0 poäng. I annat fall bedöms körningarna individuellt på följande sätt:

Låt Q vara det minsta talet sådant att det är möjligt att sortera godtycklig sekvens av sex mynt med Q vägningar på Aminas våg. För att göra uppgiften mer utmanande kommer vi inte avslöja värdet på Q.

Antag att det största antalet vägningar över *alla testfall och alla körningar* är $Q + y$ för något heltal y. Betrakta sedan en enstaka körning av ditt program. Låt det största antalet viktningar över alla T testfallen i den här körningen vara $Q + x$ för något icke-negativt heltal x. (Om du gör färre än Q viktningar för alla testfall, låt $x = 0$.) Poängen för den här körningen är då $\frac{100}{r((x+y)/5+1)}$, avrundat *nedåt* till två siffror efter decimaltecknet.

Det innebär att om programmet gör högst Q vägningar i varje testfall för varje körning kommer du att tilldelas 100 poäng.

Exempel

Antag att mynten är ordnade $3\ 4\ 6\ 2\ 1\ 5$ från lättast till tyngst.

Funktionsanrop	Returvärde	Förklaring
getMedian(4, 5, 6)	6	Mynt 6 är medianen av mynt 4, 5 och 6.
getHeaviest(3, 1, 2)	1	Mynt 1 är det tyngsta av mynt 1, 2 och 3.
getNextLightest(2, 3, 4, 5)	3	Mynt 2, 3 och 4 är alla lättare än mynt 5, så det lättaste bland dem (3) returneras.
getNextLightest(1, 6, 3, 4)	6	Mynt 1 och 6 är båda tyngre än mynt 4. Av mynt 1 och 6 så är mynt 6 det lättaste.
getHeaviest(3, 5, 6)	5	Mynt 5 är det tyngsta bland mynten 3, 5 och 6.
getMedian(1, 5, 6)	1	Mynt 1 är medianen av mynt 1, 5 och 6.
getMedian(2, 4, 6)	6	Mynt 6 är medianen av mynt 2, 4 och 6.
answer([3, 4, 6, 2, 1, 5])		Programmet hittade rätt svar för det här testfallet.

Exempelrättare

Exempelrättaren läser indata i följande format:

- rad 1: T —- antal testfall
- vardera av raderna 2 till $T + 1$: en sekvens av 6 distinkta tal från 1 till 6: ordningen på talen från lättast till tyngst.

Till exempel, ett indata bestående av två testfall där mynten är ordnade $1\ 2\ 3\ 4\ 5\ 6$ och $3\ 4\ 6\ 2\ 1\ 5$ ser ut som följer:

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

Exempelrättaren skriver ut arrayen som skickades som parameter till `answer()`-funktionen.