IOI '15 P2 - Scales

Time limit: 1.0s **Memory limit:** 256M

Amina has six coins, numbered from 1 to 6. She knows that the coins all have different weights. She would like to order them according to their weight. For this purpose she has developed a new kind of balance scale.

A traditional balance scale has two pans. To use such a scale, you place a coin into each pan and the scale will determine which coin is heavier.

Amina's new scale is more complex. It has four pans, labeled A, B, C, and D. The scale has four different settings, each of which answers a different question regarding the coins. To use the scale, Amina must place exactly one coin into each of the pans A, B, and C. Additionally, in the fourth setting she must also place exactly one coin into pan D.

The four settings will instruct the scale to answer the following four questions:

1. Which of the coins in pans A, B, and C is the heaviest?
2. Which of the coins in pans A, B, and C is the lightest?
3. Which of the coins in pans A, B, and C is the median? (This is the coin that is neither the heaviest nor the lightest of the three.)
4. Among the coins in pans A, B, and C, consider only the coins that are heavier than the coin on pan D. If there are any such coins, which of these coins is the lightest? Otherwise, if there are no such coins, which of the coins in pans A, B, and C is the lightest?

Task

Write a program that will order Amina's six coins according to their weight. The program can query Amina's scale to compare weights of coins. Your program will be given several test cases to solve, each corresponding to a new set of six coins.

Your program should implement the functions `init` and `orderCoins`. During each run of your program, the grader will first call `init` exactly once. This gives you the number of test cases and allows you to initialize any variables. The grader will then call `orderCoins()` once per test case.

```
void init(int T)
```

- `T`: The number of test cases your program will have to solve during this run, such that $1 \leq T \leq 18$.

```
void orderCoins(void)
```

- This function is called exactly once per test case.
- The function should determine the correct order of Amina's coins by calling the grader functions `getHeaviest()`, `getLightest()`, `getMedian()`, and/or `getNextLightest()`.

- Once the function knows the correct order, it should report it by calling the grader function `answer()`.
- After calling `answer()`, the function `orderCoins()` should return.

Grader Functions

```
void answer(int W[6])
```

- `W`: An array of length 6 containing the correct order of coins. `W[0]` through `W[5]` should be the coin numbers (i.e., numbers from 1 to 6) in order from the lightest to the heaviest coin.
- Your program should only call this function from `orderCoins()`, once per test case.

```
int getHeaviest(int A, int B, int C)
int getLightest(int A, int B, int C)
int getMedian(int A, int B, int C)
```

- These correspond to settings 1, 2 and 3 respectively for Amina's scale.
- `A`, `B`, `C`: The coins that are put in pans A, B, and C, respectively. A, B, and C should be three distinct integers, each between 1 and 6 inclusive.
- Each function returns one of the numbers `A`, `B`, and `C`: the number of the appropriate coin. For example, `getHeaviest(A, B, C)` returns the number of the heaviest of the three given coins.

```
int getNextLightest(int A, int B, int C, int D)
```

- This corresponds to setting 4 for Amina's scale.
- `A`, `B`, `C`, `D`: The coins that are put in pans A, B, C, and D, respectively. `A`, `B`, `C`, and `D` should be four distinct integers, each between 1 and 6 inclusive.
- The function returns one of the numbers `A`, `B`, and `C`: the number of the coin selected by the scale as described above for setting 4. That is, the returned coin is the lightest amongst those coins on pans A, B, and C that are heavier than the coin in pan D; or, if none of them is heavier than the coin on pan D, the returned coin is simply the lightest of all three coins on pans A, B, and 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.