Balanzas

Amina tiene seis monedas numeradas de 1 a 6. Sabe que las monedas tienen todas un peso distinto y quiere ordenarlas segun su peso. Para ello ha desarrollado un nuevo tipo de balanza de platos.

La balanza tradicional tiene dos platos. Para usar esta balanza debes colocar una moneda en cada plato y la balanza te dirá cual pesa más.

La nueva balanza de Amina es más compleja. Tiene cuatro platos etiquetados: A, B, C, y D. La balanza tiene 4 configuraciones distintas, cada una con respuestas a diferentes preguntas sobre las monedas. Para usarla, Amina tiene que colocar exactamente una moneda en cada uno de los platos A, B, C, y para la cuarta configuración tiene que colocar también una única moneda en el plato D. Las cuatro configuraciones responden a las siguientes 4 preguntas:

1. ¿Cuál de las monedas en los platos A, B y C es la más pesada?
2. ¿Cuál de las monedas en los platos A, B y C es la más ligera?
3. ¿Cuál de las monedas en los platos A, B y C es la mediana? (Es decir, la moneda que no es la más ligera ni la más pesada de las tres)
4. Entre las monedas en los platos A, B, y C, considera solo las monedas que son más pesadas que la moneda en el plato D. Si hay alguna de estas monedas, ¿cuál es de estas la más ligera? Si no hay ninguna moneda más pesada que la moneda del plato D, ¿cuál de las monedas en los platos A, B y C es la más ligera?

El problema

Escribe un programa que ordene las seis monedas de Amina según su peso. El programa puede hacer preguntas a la balanza de Amina para comparar los pesos de las monedas. Tu programa recibirá varios casos para resolver, cada caso corresponde a un nuevo conjunto de seis monedas.

Tu programa deberá implementar las funciones `init` y `orderCoins`. En cada ejecución de tu programa, el grader llamará primero a `init` exactamente una vez. Esto te dará el número de juegos de prueba y te permitirá inicializar las variables. El grader llamará entonces a `orderCoins()` una vez por caso.

- `init(T)`
  - `T`: El número de casos de pruebas que tu programa debe resolver en esta ejecución. `T` es un entero en $1, \ldots, 18$.
  - Esta función no devuelve ningún valor.
- `orderCoins()`
  - A esta función se le llama exactamente una vez por caso de prueba.

- La función debe determinar el orden correcto de las monedas de Amina usando las funciones del grader `getHeaviest()`, `getLightest()`, `getMedian()`, y/o `getNextLightest()`.
- Una vez la función sepa el orden correcto, tiene que devolverlo usando la función del grader `answer()`.
- Despues de realizar la llamada `answer()`, la función `orderCoins()` debe acabar sin devolver ningún valor.

Puedes usar las siguientes funciones del grader en tu programa:

- `answer(W)` — tu programa debe usar esta función para devolver el resultado que ha encontrado.
  - `W`: Un array de tamaño 6 que contenga el orden correcto de las monedas. Las posiciones entre `W[0]` y `W[5]` deberán ser los números de las monedas (i.e., números del 1 al 6) en orden de la más ligera a la más pesada.
  - Tu programa solo deberá llamar a esta función desde `orderCoins()`, una vez por juego de pruebas.
  - Esta función no devuelve ningún valor.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — estas funciones corresponden a las configuraciones 1, 2 y 3 respectivamente para las balanzas de Amina.
  - `A`, `B`, `C`: Las monedas que se colocan en los platos A, B, y C, respectivamente. `A`, `B`, y `C` tienen que ser tres enteros distintos entre 1 y 6 incluidos.
  - Esta función devuelve uno de los números `A`, `B`, o `C`: el número de la moneda correspondiente. Por ejemplo, `getHeaviest(A, B, C)` devuelve el número de la moneda más pesada de entre las 3.
- `getNextLightest(A, B, C, D)` — esta función corresponde a la configuración 4 para las balanzas de Amina.
  - `A`, `B`, `C`, `D`: Las monedas en los platos A, B, C, y D, respectivamente. `A`, `B`, `C`, y `D` deberían ser cuatro enteros distintos, cada uno entre 1 y 6 inclusivos.
  - La función devuelve uno de los números `A`, `B`, y `C`: el número de la moneda seleccionada por la balanza con el critério anteriormente descrito en la configuración 4. Esto es, la moneda más ligera entre las monedas en los platos que sean más pesadas que la moneda en el plato D, y si ninguna de ellas es más pesada que la moneda en el plato D entonces devuelve la moneda más ligera entre las tres que se encuentran en los platos A, B, y C.

Puntuación

No hay subtareas en este problema. En su lugar, tu puntuación se basará en cuántas pesadas realices (número total de llamadas a las funciones del grader `getLightest()`, `getHeaviest()`, `getMedian()` y/o `getNextLightest()`).

Tu programa se ejecutará varias veces con varios juegos de prueba en cada ejecución. Sea r el

número de veces que se debe ejecutar tu programa (este número esta fijado en la entrada). Si tu programa no ordena las monedas correctamente en alguno de los juegos de prueba, tu programa recibirá 0 puntos. Si no, los programas se evalúan como sigue:

Sea Q el número mínimo tal que es posible ordenar cualquier secuencia de seis monedas usando Q pesadas en la balanza de Amina. Para hacer la tarea más interesante no te diremos cual es el valor de Q.

Supón que el número mas grande de pesadas entre todos los casos de prueba de todas las ejecuciones es $Q + y$ para un entero y. Entonces, considera una ejecución en particular. Sea $Q + x$ el número más grande de pesadas entre todos los T casos en esta ejecución para un entero no negativo x (si usas menos de Q pesadas para cada caso de prueba, entonces $x = 0$.) Entonces, la puntuación de esta ejecución es $\frac{100}{r((x+y)/5+1)}$, redondeada a la baja a dos decimales.

En particular, si tu programa hace como mucho Q pesadas en cada caso de cada ejecución, tendrás 100 puntos.

Ejemplo

Supon que las monedas estan ordenadas: $3\ 4\ 6\ 2\ 1\ 5$ de más ligera a más pesada.

Llamada a la función	Returns	Explicación
getMedian(4, 5, 6)	6	La moneda 6 es la mediana entre las monedas 4, 5, y 6.
getHeaviest(3, 1, 2)	1	La moneda 1 es la más pesada entre las monedas 1, 2, y 3.
getNextLightest(2, 3, 4, 5)	3	Las monedas 2, 3, y 4 son todas más ligeras que la moneda 5, entonces la más ligera entre ellas es devuelta (3).
getNextLightest(1, 6, 3, 4)	6	Las monedas 1 y 6 son más pesadas que la moneda 4. Entre estas monedas (1 y 6), la moneda 6 es la más ligera.
getHeaviest(3, 5, 6)	5	La moneda 5 es la más pesada de entre las monedas 3, 5 y 6.
getMedian(1, 5, 6)	1	La moneda 1 es la mediana entre las monedas 1, 5 y 6.
getMedian(2, 4, 6)	6	La moneda 6 es la mediana entre las monedas 2, 4 y 6.
answer([3, 4, 6, 2, 1, 5])		El programa ha encontrado la respuesta correcta a este caso de pruebas.

Sample grader

El sample grader lee el input en el formato siguiente:

- line 1: T —- número de casos de prueba
- cada una de las líneas de la 2 hasta la $T + 1$: una secuencia de 6 números distintos de 1 a 6: el orden de las monedas de menor a mayor peso.

Por ejemplo, una entrada formada por dos casos de prueba donde las monedas estan ordenadas $1\ 2\ 3\ 4\ 5\ 6$ y $3\ 4\ 6\ 2\ 1\ 5$ se vería como:

```
2
```

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

El evaluador de muestra imprime el array que se le pasa como parametro a la función `answer()`.