Balanza

Amina tiene seis monedas, numeradas de 1 a 6. Ella sabe que cada moneda tiene un peso diferente. A ella le gustaría ordenar las monedas de acuerdo a su peso. Para esto ella ha desarrollado un nuevo tipo de balanza.

Una balanza tradicional tiene dos platos. Para usar este tipo de balanza, debes poner una moneda en cada plato y la balanza determinará cuál moneda es la más pesada.

La nueva balanza de Amina es más compleja. Esta posee cuatro platos, etiquetados A, B, C y D. La balanza tiene cuatro configuraciones diferentes, cada una para responder una pregunta diferente. Para usar esta balanza, Amina debe poner exactamente una moneda en cada uno de los platos A, B, y C. Adicionalmente, sólo en la cuarta configuración, ella debe poner una moneda en el plato D.

Cada una de las cuatro configuraciones responderá una de las siguientes cuatro 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 liviana?
3. ¿Cuál de las monedas en los platos A, B, y C es la mediana? (Esta es la moneda que no es ni la más pesada ni la más liviana.)
4. Entre las monedas en los platos A, B, y C, considera las monedas que son más pesadas que la moneda en el plato D. Si hay monedas que cumplen esto, ¿cuál de estas es la más liviana? En caso contrario, si ninguna de las monedas en A, B, y C es más pesada que la moneda en D, responde ¿cuál de las monedas en los platos A, B, y C es la más liviana?

Tarea

Escribe un programa que ordene las seis monedas de Amina de acuerdo a su peso. El programa puede hacer preguntas a la balanza de Amina para comparar el peso de las monedas. Tu programa será probado con varios casos de prueba, cada uno correspondiente a un nuevo conjunto de seis monedas.

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

- `init(T)`
  - `T`: El número de casos de prueba que tu programa deberá resolver durante esta ejecución. `T` es un entero en el rango $1, \ldots, 18$.
  - Esta función no tiene valor de retorno.
- `orderCoins()`

- Esta función es llamada exactamente una vez por cada caso de prueba.
- La función debe determinar el orden correcto para las monedas de Amina llamando a las funciones `getHeaviest()`, `getLightest()`, `getMedian()`, y/o `getNextLightest()`.
- Una vez que la función determina el orden correcto debe reportarlo, llamando a la función `answer()`.
- Después de llamar a `answer()`, la función `orderCoins()` debe retornar. `orderCoins()` no tiene ningún valor de retorno.

Puedes usar las siguientes funciones en tu programa:

- `answer(W)` — tu programa debe usar esta función para reportar la respuesta que encuentre.
  - `W`: Un arreglo de tamaño 6 que contiene el orden correcto de las monedas. `W[0]` hasta `W[5]` debe ser el número de cada moneda (es decir, números de 1 a 6) en el orden de la más liviana a la más pesada.
  - Tu programa debe llamar esta función desde `orderCoins()` solo una vez por caso de prueba.
  - Esta función no tiene valor de retorno.
- `getHeaviest(A, B, C)`, `getLightest(A, B, C)`, `getMedian(A, B, C)` — estas corresponden respectivamente a las configuraciones 1, 2 y 3 en la balanza de Amina.
  - `A`, `B`, `C`: Las monedas para poner en los platos A, B, y C, respectivamente. `A`, `B`, y `C` deben ser tres enteros distintos entre 1 y 6.
  - Cada función retorna uno de los números `A`, `B`, y `C`: el número de la moneda apropiada. Por ejemplo, `getHeaviest(A, B, C)` retorna el número de la moneda más pesada entre las tres dadas como parámetro.
- `getNextLightest(A, B, C, D)` — esta corresponde a la configuración 4 en la balanza de Amina.
  - `A`, `B`, `C`, `D`: Las monedas para poner en los platos A, B, C, y D, respectivamente. `A`, `B`, `C`, y `D` deben ser cuatro enteros distintos entre 1 y 6.
  - La función retorna uno de los números `A`, `B`, y `C`: el número de la moneda seleccionada por la balanza de la forma descrita para la configuración 4. Esto es, la moneda retornada es la más liviana entre aquellas monedas en los platos A, B, C que son más pesadas que la moneda en el plato D; o, si ningúna es más pesada que la moneda en el plato D, la moneda retornada es simplemente la más liviana entre las monedas en los platos A, B, C.

Puntaje

No hay subtareas en este problema. En vez de esto, tu puntaje se basará en cuántas veces tu programa usa la balanza (el número total de llamadas a las funciones `getLightest()`, `getHeaviest()`, `getMedian()` y/o `getNextLightest()`).

Tu programa se ejecutará varias veces con multiples casos de prueba en cada ejecución. Definamos

como r el número de ejecuciones de tu programa. Este número está fijado por los casos de prueba. Tu programa obtendrá un puntaje mayor que cero solo si ordena correctamente cada caso de prueba de cada ejecución. En ese caso, cada ejecución será evaluada individualmente como se describe a continuación.

Sea Q el menor número con el cuál es posible ordenar la secuencia de seis monedas usando Q veces la balanza. Para hacer la tarea más desafiante, no revelaremos el valor de Q aquí.

Supón que el número máximo de usos de la balanza entre todos los casos de prueba de todas las ejecuciones es $Q + y$ para algún entero y. Considera ahora una sola ejecución de tu programa. Sea $Q + x$ el número más grande de usos de la balanza entre todos los T casos de prueba en una sóla ejecución para algún entero no negativo x. (Si usas menos de Q usos de la balanza para cada caso de prueba, entonces $x = 0$.) Entonces, el puntaje para esa ejecución será $\frac{100}{r((x+y)/5+1)}$, truncado a dos dígitos después del punto decimal.

En particular, si tu programa realiza a lo más Q usos de la balanza en cada caso de prueba para cada ejecución, obtendrás 100 puntos.

Ejemplo

Supón que las monedas están ordenadas $3\ 4\ 6\ 2\ 1\ 5$ desde la más liviana a la más pesada.

Llamada a función	Valor de retorno	Explicación
getMedian(4, 5, 6)	6	La moneda 6 es la media 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 más livianas que la moneda 5, luego la más liviana entre ellas (3) es retornada.
getNextLightest(1, 6, 3, 4)	6	Las monedas 1 y 6 son ambas más pesadas que la moneda 4. Entre las monedas 1 y 6, la moneda 6 es la más liviana.
getHeaviest(3, 5, 6)	5	La moneda 5 es la más pesada 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 encontró la respuesta correcta para este caso de prueba.

Grader de ejemplo

El grader de ejemplo lee el input en el siguiente formato:

- línea 1: T —- El número de casos de prueba
- cada una de las líneas de la 2 a la $T + 1$: una secuencia de 6 números distintos entre 1 y 6: el orden de las monedas desde las más liviana hasta la más pesada.

Por ejemplo, un input que consiste en dos casos de prueba donde el orden de las monedas es $1\ 2\ 3\ 4\ 5\ 6$ y $3\ 4\ 6\ 2\ 1\ 5$ se ve de la siguiente forma:

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

El grader de ejemplo imprime el arreglo que es pasado como parametro a la función `answer()`.