Soluciones entrenos OIE

Cluedo

El Dr. Black ha sido asesinado. La detective Jill debe determinar el asesino, la ubicación y el arma. Hay seis posibles asesinos, numerados del 1 al 6. Hay diez ubicaciones posibles, numeradas del 1 al 10. Hay seis armas posibles, numeradas del 1 al 6.

Jill intenta repetidamente adivinar la combinación correcta de asesino, ubicación y arma. Cada intento se llama teoría. Ella le pide a su asistente Jack que confirme o refute sus teorías una a una. El juego se termina en cuanto Jack le confirme a Jill que su última teoría es la correcta. Cuando Jack refuta una teoría, le informa a Jill que el asesino, la ubicación o el arma es incorrecto, diciendo cuál de ellos es.

Detalles de implementación

Debes implementar la siguiente función:

void Solve()

- Esta función es llamada exactamente una vez para cada caso de prueba.
- Esta función debe llamar repetidamente a **Theory(M, L, W)**, que está ya implementada como parte del grader. M, L y W son números que indican una combinación particular de asesino, ubicación y arma.
- La función debe finalizar su ejecución cuando al llamar a la función **Theory(M, L, W)**, esta devuelva el valor 0.

Puedes llamar a la siguiente función, que está ya implementada por el grader:

int Theory(int M**, int** L**, int** W**)**

- M, L, W representan el número del asesino, del lugar y del arma, respectivamente.
- Se han de cumplir las siguientes condiciones: $1 \leq M \leq 6$, $1 \leq L \leq 10$ y $1 \leq W \leq 6$. Si alguna de estas condiciones no se cumple, se finalizará la ejecución del programa.
- Si la teoría es correcta la función devuelve el valor 0.
- Si la teoría es incorrecta, se devuelve un valor de entre 1, 2 o 3. 1 indica que el asesino es incorrecto; 2 indica que la ubicación es incorrecta; 3 indica que el arma está mal. Si más de uno está equivocado, Jack elige uno arbitrariamente entre los incorrectos (no necesariamente de una manera determinista).

Nota importante: Cualquier variable utilizada por **Solve**, ya sea global o local, se reinicializará cada vez que el grader llame a **Solve**.

Ejemplo

Pongamos que en esta situación, la solución es $M = 2$, $L = 3$, $W = 4$. Estos son los valores devueltos en varias llamadas posibles de **Theory(M, L, W)**.

Llamada	Valor devuelto	Explicación
Theory(1, 1, 1)	1, 2 o 3	Los tres son incorrectos
Theory(3, 3, 3)	1 o 3	Solo el lugar es correcto
Theory(5, 3, 4)	1	Solo el asesino está mal
Theory(2, 3, 4)	0	La teoría es correcta

Constraints

El grader llamará a **Solve** hasta 100 veces.

Subtareas

1. (50 puntos) Cada llamada a **Solve** puede llamar a **Theory(M, L, W)** un máximo de 360 veces.
2. (50 puntos) Cada llamada a **Solve** puede llamar a **Theory(M, L, W)** un máximo de 20 veces.

Solución

La solución más sencilla consiste en probar todas las combinaciones posibles hasta que alguna de ellas sea correcta. Podríamos, por ejemplo, utilizar tres ciclos for de la siguiente forma:

```
void Solve(){
   for(int M = 1; M <= 6; M++) {
      for(int L = 1; L <= 10; L++) {
         for(int W = 1; W <= 6; W++) {
            int respuesta = Theory(M, L, W);
            if(respuesta == 0) {
               return;
            }
         }
      }
   }
}
```

En total hay $6 * 10 * 6 = 360$ combinaciones que comprobar. Si tenemos suerte, encontraremos la combinación correcta con pocas operaciones. Pero si la combinación correcta es (6, 10, 6), entonces habremos comprobado 359 combinaciones incorrectas antes de llegar a la combinación adecuada, por lo que en total habremos efectuado 360 llamadas a **Theory(M, L, W)**. En cualquier caso, haremos un máximo de 360 llamadas para llegar a la combinación correcta, por lo que esta solución es suficiente para resolver la primera subtarea.

Para resolver la segunda subtarea, tenemos que utilizar la información que nos devuelve la función **Theory(M, L, W)** al llamarla con una combinación incorrecta. Esta función nos indica al menos un valor incorrecto de los 3 que hemos utilizado para llamarla. Si por ejemplo efectuamos la llamada **Theory(1, 1, 1)** y la función nos devuelve el valor 1, sabemos que el asesino no tiene valor igual a 1, por lo que podemos descartar todas las combinaciones en las cuales el parámetro M sea 1.

Una forma de poder descartar valores que sabemos que van a ser incorrectos es ir incrementando los valores M, L y W empezando con la combinación $M = 1$, $L = 1$ y $W = 1$. Si nos dice, por ejemplo, que el asesino no es 1, incrementamos M. Hacemos lo mismo con el resto de valores. Si mediante este proceso llegamos por ejemplo a la combinación $(4, 4, 4)$, sabemos que cualquier combinación, en la cual alguno de los valores sea menor de 4, es incorrecta.

De esta forma realizaremos como mucho $5 + 9 + 5 = 19$ operaciones antes de llegar a la correcta. Así que en cualquier caso efectuaremos como mucho 20 llamadas a la función **Theory(M, L, W)**.

Código

C++

```cpp
void Solve(){
    int M = 1; //asesino
    int L = 1; //lugar
    int W = 1; //arma

    while(true) {
        //comprobamos si la combinación actual (M, L, W) es correcta
        int respuesta = Theory(M, L, W);

        if(respuesta == 0) {
            //la combinación es correcta
            return;
        } else if(respuesta == 1) {
            //M no es el asesino correcto
            //a partir de ahora probamos con el siguiente valor
            M++;
        } else if(respuesta == 2) {
            //L no es el lugar correcto
            //a partir de ahora probamos con el siguiente valor
            L++;
        } else if(respuesta == 3) {
            //W no es el arma correcta
            //a partir de ahora probamos con el siguiente valor
            W++;
        }
    }
}
```