Ciclo 1 Fundamentos de programación

Reto 4

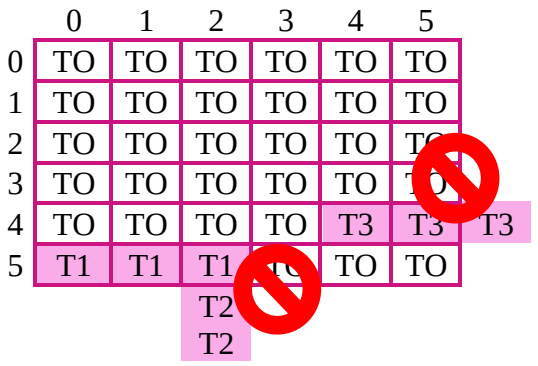
Descripción del problema: Un diseñador de videojuegos desea lanzar su primer prototipo de un juego de tablero de dos personas. Cada jugador debe ubicar en su tablero tres tesoros, y cada tesoro tendrá un espacio de tres casillas. El juego consiste en adivinar los tesoros de cada jugador, teniendo en cuenta que el jugador se lleva el tesoro completo cuando complete las tres casillas ya sea vertical u horizontal del jugador oponente.

Como primer paso para el diseño del videojuego el diseñador debe permitirle a cada jugador establecer sus tesoros en un tablero de dimensiones 6 x 6; 36 casillas en total. Las casillas que contienen el tesoro se establecerán con las letras 'T1', 'T2' y 'T3' y las casillas que no tienen tesoro se establecerán con la letra 'TO'. Se debe tener en cuenta que un tesoro no puede sobreponerse sobre otro tesoro, así como se muestran en las pruebas siguientes:

Prueba 1

	0	1	2	3	4	5
0	T1	T1	T1	TO	TO	TO
1	TO	TO	TO	TO	TO	TO
2	TO	TO	TO	TO	TO	TO
3	T3	TO	TO	T2	T2	T2
4	T3	TO	TO	TO	TO	TO
5	T3	TO	TO	TO	TO	TO

Prueba 2



El tablero de Prueba 1 está correcto, debido a que los tres tesoros están en ubicaciones distintas y no se sobreponen, tampoco se salen de las dimensiones del tablero 6 x 6. La prueba 2, tiene tres (3) errores, el primer error corresponde a un tesoro que se sobrepone en el otro 'T2' con 'T1', el segundo error corresponde al segundo tesoro 'T2' que se sale de las dimensiones del tablero, y el tercer error corresponde al tercer tesoro 'T3' que se sale de las dimensiones del tablero.

Nota: Recuerde métodos: map, filter, lambda, zip o reduce para desarrollar el ejercicio

Desarrolle un programa que permita establecer **un** (1) tablero con las ubicaciones de los tesoros ‘T1’, ‘T2’ y ‘T3’ correctamente. Para esto debe seguir las siguientes reglas:

1. El tablero inicial es de 6 x 6 y tiene las letras ‘TO’, ósea, tablero en blanco.

	0	1	2	3	4	5
0	TO	TO	TO	TO	TO	TO
1	TO	TO	TO	TO	TO	TO
2	TO	TO	TO	TO	TO	TO
3	TO	TO	TO	TO	TO	TO
4	TO	TO	TO	TO	TO	TO
5	TO	TO	TO	TO	TO	TO

2. La información de entrada está en un **diccionario** que tiene dos llaves de primer nivel denominadas **‘nick’** y **‘ubicaciones’**. La llave **‘ubicaciones’** contiene tres diccionarios con llaves **‘T1’**, **‘T2’** y **‘T3’**, cada llave de segundo nivel tiene como valor una lista cuyas dos primeras posiciones corresponden a la posición de la fila – columna respectivamente y la tercera posición corresponde a la dirección ‘D’ si es Derecha o ‘A’ si es Abajo.

   Ejemplo:

```
diccionario = {'nick':'lucila',
               'ubicaciones: { 'T1': [0, 0, 'D'],
                               'T2': [3, 0, 'A'],
                               'T3': [3, 3, 'D'],
                              }
              }
```

3. Para finalizar cada tesoro se genera automáticamente partir de las indicaciones anteriores, es decir si un tesoro comienza en la posición 0,0,D significa que desde la posición 0,0 hacía la derecha se generará, como se mostró en la *Prueba 1*.

Usted debe elaborar una función que reciba como parámetro un **diccionario** como el descrito en el **punto 2** y retorne un diccionario que contenga las siguientes llaves **‘nombreJugador’**, **‘tesorosErroneos’** y **‘tablero’**. La llave ‘nombreJugador’ corresponde al Nick del jugador
La llave ‘tesorosErroneos’ puede contener dos valores, si no hay error al poner los tesoros será 0, de lo contrario será una lista de los nombres de los tesoros con error y la llave ‘tablero’ si no hay error será una lista compuesta (lista de lista, o matriz) con los tesoros ubicados, de lo contrario será -1.

Ejemplo

Entrada	Salida
Información: dict	**return**

informacionPrueba1 = {
'nick':'lucila',
'ubicaciones':
{
'T1':[0,0,'D'],
'T2':[3,0,'A'],
'T3':[3,3,'D'],
}
}
	{
	'nombreJugador': 'lucila',
	'tesorosErroneos': 0,
	'tablero': [['T1', 'T1', 'T1', 'TO', 'TO', 'TO'],
	['TO', 'TO', 'TO', 'TO', 'TO', 'TO'],
	['TO', 'TO', 'TO', 'TO', 'TO', 'TO'],
	['T2', 'TO', 'TO', 'T3', 'T3', 'T3'],
	['T2', 'TO', 'TO', 'TO', 'TO', 'TO'],
	['T2', 'TO', 'TO', 'TO', 'TO', 'TO']]
	}

informacionPrueba2 = {
'nick':'juanito',
'ubicaciones':
{
'T1':[5,0,'D'],
'T2':[5,2,'A'],
'T3':[4,4,'D'],
}
}
	{'nombreJugador': 'juanito', 'tesorosErroneos': ['T2', 'T3'], 'tablero': -1}

informacionPrueba3 = {
'nick':'paco',
'ubicaciones':
{
'T1':[-1,0,'D'],
'T2':[0,2,'A'],
'T3':[4,4,'D'],
}
}
	{'nombreJugador': 'paco', 'tesorosErroneos': ['T1', 'T3'], 'tablero': -1}

Entrada: Diccionario que contiene las siguientes llaves:

Nombre	Tipo	Descripción
nick	str	Nick del jugador
ubicaciones	dict	Llave que contiene tres diccionarios correspondientes a cada tesoro ‘T1’, ‘T2’ y ‘T3’
T1, T2 y T3	dict	Llaves de segundo nivel, cuyo valor es una lista, donde la posición 0 será la posición horizontal del tesoro, la posición 1 será la posición vertical del tesoro y la posición 2 será la dirección del tesoro, ‘D’ derecha o ‘A’ abajo.

Salida: Retorno de un diccionario que contiene las siguientes llaves:

Nombre	Tipo	Descripción
nombreJugador	str	Nick del jugador
tesorosErroneos	int / list	Puede tomar dos valores, si no hay error al poner los tesoros será 0, de lo contrario será una lista de los nombres de los tesoros con error
tablero	list / int	si no hay error será una lista compuesta (lista de lista, o matriz) con los tesoros ubicados, de lo contrario será -1

Esqueleto:

```
def establecerTablero(informacion:dict) -> dict:

    pass
```