PECL2

Se quiere hacer un programa en Python para gestionar los rankings de usuarios en una Web de speedrunning. En estas Webs, los jugadores juegan a videojuegos populares y registran el tiempo que han tardado en terminarlos. Con esa información se hacen rankings de los jugadores más rápidos para cada juego. Un ejemplo es este: https://www.speedrun.com/sm64

Para ello, se representan las partidas jugadas mediante un diccionario, cuyas claves son los nombres de los usuarios, y cuyos valores son a su vez un diccionario que contiene la descripción del jugador y una lista de partidas jugadas.

El siguiente ejemplo representa la información de dos jugadores.

```
{"pepe": {
        "desc": "Soy pepe y mi juego preferido es MINECRAFT.",
        "partidas" : [("amongus", 12.5), ("minecraft", 24.2),
("minecraft", 28.2), ("supermario odissey", 45.5)]
        },
 "juan": {
        "desc": "Me llamo juan y me gusta MINECRAFT y AMONGUS.",
        "partidas": [("portal2", 45.0), ("amongus", 10.2)]
       }
 }
```

Nótese que un mismo jugador puede jugar varias veces al mismo juego, obteniendo diferentes tiempos, y cada jugador juega a los juegos que quiere.

Un programa comienza con una base de datos de jugadores vacia, llamando a la siguiente función que simplemente nos devuelve un diccionario vacío:

```
def crea_bd_jugadores():
    """Crea una base de datos de jugadores vacia"""
    return {}
```

Siguiendo la representación indicada, se pide escribir en Python funciones para hacer lo siguiente:

Incluir partidas (2 puntos)

- Añadir una partida de un usuario. Para ello, se programará una función con la siguiente cabecera:

```
def incluir_partida(bd, jugador:str, desc: str, juego: str, tiempo:
float):
    """Incluye una nueva partida en la base de datos.
       Si es la primera partida del jugador, hay que incluirle,
       si el jugador ya tenía partidas, se actualiza la información,
       sustituyendo su descripción y añadiendo la partida a la
       lista de partidas ya existente.
    """
```

Por ejemplo,

```
bd = crea_bd_jugadores()
incluir_partida(bd, "juan", "Me llamo juan y me gusta MINECRAFT y
AMONGUS", "portal2", 45.0)
```

```
incluir_partida(bd, "juan", "Me llamo Juan y me gusta mucho MINECRAFT
y AMONGUS", "amongus", 10.2)
incluir_partida(bd, "juan", "Me llamo Juan y me gusta mucho MINECRAFT
y AMONGUS", "amongus", 100.2)
incluir_partida(bd, "pepe", "Soy pepe y mi juego preferido es
MINECRAFT", "amongus", 12.5)
incluir_partida(bd, "ana", "Soy ana y suelo jugara a MINECRAFT",
"minecraft", 20.3)
incluir_partida(bd, "ana", "Soy ana y suelo jugara a MINECRAFT",
"minecraft", 22.3)
incluir_partida(bd, "ana", "Soy ana y suelo jugara a MINECRAFT",
"amongus", 22.3)

print(bd)
```

La salida del código anterior sería la siguiente.

```
{'juan': {'partidas': [('portal2', 45.0), ('amongus', 10.2),
('amongus', 100.2)], 'desc': 'Me llamo Juan y me gusta mucho MINECRAFT
y AMONGUS'}, 'pepe': {'partidas': [('amongus', 12.5)], 'desc': 'Soy
pepe y mi juego preferido es MINECRAFT'}, 'ana': {'partidas':
[('minecraft', 20.3), ('minecraft', 22.3), ('amongus', 22.3)], 'desc':
'Soy ana y suelo jugara a MINECRAFT'}}
```

El juego más popular (3 puntos)

- Obtener el juego que se ha jugado más veces de los que ya están en la base de datos y el número de partidas jugadas de ese juego.

```
def obtener_juego_mas_frecuente(bd)-> (str, int):
    """Obtiene el juego del que se han jugado más partidas.
    """
```

Por ejemplo, siguiendo el ejemplo con los datos anteriores:

```
print(obtener_juego_mas_frecuente(bd))
```

El resultado sería el siguiente.

```
('amongus', 4)
```

Hacer hashtags (2 puntos)

- Los jugadores en sus descripciones a veces mencionan juegos, escribiendo sus nombres en mayúsculas. Se quiere hacer una función que identifique esos juegos en las descripciones y los convierta a hastags.

```
def crear_hashtags(bd):
    """Identifica nombres de juegos en mayúsculas en las descripciones
de los jugadores
       y los convierte en hashtags.
    """
```

Siguiendo el ejemplo con los datos anteriores:

```
crear_hashtags(bd)
print(bd)
```

El resultado sería el siguiente.

```
{'juan': {'partidas': [('portal2', 45.0), ('amongus', 10.2),
('amongus', 100.2)], 'desc': 'Me llamo Juan y me gusta mucho
#MINECRAFT y #AMONGUS'}, 'pepe': {'partidas': [('amongus', 12.5)],
'desc': 'Soy pepe y mi juego preferido es #MINECRAFT'}, 'ana':
{'partidas': [('minecraft', 20.3), ('minecraft', 22.3), ('amongus',
22.3)], 'desc': 'Soy ana y suelo jugara a #MINECRAFT'}}
```

Ranking de jugadores (3 puntos)

- Obtener el ranking de los M mejores jugadores para un juego determinado, teniendo en cuenta que es mejor cuanto más bajo es el tiempo.

Nótese que un mismo jugador solo puede aparecer una vez en el ranking, por lo que para cada jugador hay que coger **su mejor tiempo**, dado que un jugador puede haber jugado varias veces al juego, con diferentes tiempos.

```
def obtener_ranking(bd, juego:str, m: int)->list:
    """Obtiene el ranking de los `m` mejores jugadores del juego.
    """
```

Siguiendo el ejemplo anterior.

```
print(obtener_ranking(bd, "amongus", 3))
print(obtener_ranking(bd, "amongus", 2))
```

El resultado sería el siguiente.

```
[('juan', 10.2), ('pepe', 12.5), ('ana', 20.3)]
[('juan', 10.2), ('pepe', 12.5)]
```

Buscar recursivamente (2 puntos)

- Utilizando búsquedas recursivas, determinar si un jugador ha jugado o no a un determinado juego. NOTA: no se puede utilizar el operador `in` de Python para determinar inclusión en una lista.

Nótese que hay que comprobar que el jugador está en la base de datos.

```
def ha_jugado(db, jugador: str, juego: str)->bool:
    """Devuelve verdadero si el jugador ha jugado al juego al menos
una vez.
    """
```

Siguiendo el ejemplo anterior.

```
print(ha_jugado(bd, "juan", "amongus"))
print(ha_jugado(bd, "juan", "Garrys Mod"))
print(ha_jugado(bd, "antonio", "rdd"))
```

El resultado sería el siguiente.

```
True
False
False
```