**Tema 3 [45%]**

El acumula-tesoros es un juego de tablero en el que un jugador debe atravesar una ciudad avanzando y retrocediendo por un corredor de 50 metros de largo. Mientras atraviesa la ciudad, al jugador se le asigna, aleatoriamente, al llegar a cada posición uno de los siguientes ***estados***: fortaleza (-1), hambre (-2) y súper-héroe (-3). Algunos metros pueden tener asignados uno de los siguientes ***elementos***: armas (1), víveres (2) o tesoros (3), elementos que podrán ser recogidos si el jugador está en el estado adecuado:

 Para recoger armas, el jugador debe estar en el estado de Fortaleza.

 Para recoger víveres, el jugador debe estar en el estado de Hambre.

 Para recoger tesoros, el jugador debe estar en el estado de Súper-Héroe.

Cada jugador avanza usando un dado de 6 caras y en cada movimiento se verifica:

 Si la ubicación a donde se mueve el jugador es un numero primo entonces se debe retroceder 10 metros o volver al inicio si estuviera a menos de 10 metros del mismo.

 Caso contrario, el jugador puede recoger algún elemento del camino dependiendo del estado en el que se encuentre.

Al inicio del juego, usted debe distribuir aleatoriamente 8 armas, 8 víveres y 8 tesoros a lo largo del corredor, ***excepto*** en la posición 1, en donde se ubica el jugador al iniciar la partida. El juego termina cuando el jugador llega al final del corredor **(en este caso GANA),** o si el jugador ha caído 3 veces en una posición que es un número primo **(en este caso PIERDE).** Al finalizar se debe calcular la riqueza que ha ganado el jugador, la misma que depende de los elementos recogidos a lo largo del juego y se calcula mediante la siguiente fórmula:

**Riqueza** = (armas/2+1)\*100 + víveres\*0.5 + tesoros

A usted se le solicita implementar en Python:

1. La función ***asignarPosicionElementos*** que retorna una colección que contiene 3 listas, cada una representando las ***posiciones*** en el corredor de las 8 armas, víveres y tesoros, respectivamente.

2. La función ***lanzarDado*** que retorna aleatoriamente el valor de una de las ***caras*** del dado.

3. La función ***generarEstado*** que retorna aleatoriamente uno de los posibles ***estados***.

4. La función ***mostrarElemento*** que dada una ***posición*** y la colección de ***posiciones de los elementos*** imprime en pantalla el tipo de elemento que existe en esa posición.

5. La función ***recogerElemento*** que dada una posición, el estado de un jugador y la colección de ***posiciones de los elementos*** retorna el ***tipo de elemento*** que se ha recogido (1, 2 ó 3) o 0 si no ha sido posible.

6. La función ***mostrarMensaje***, la cual recibe como parámetros el ***estado*** del jugador y su ***posición*** para imprimirlos en pantalla.

7. La función ***calcularRiqueza*** que dada una cantidad de ***armas***, ***víveres*** y ***tesoros*** retorna la ***riqueza*** obtenida.

8. Un programa que simule el juego y **use** las funciones implementadas anteriormente.

Al finalizar el juego, se debe declarar al ganador y la riqueza de cada uno.

En el turno de cada jugador, se deberá mostrar por pantalla:

 El resultado del lanzamiento del dado y la nueva posición del jugador

 Si hay un retroceso

Escriba una función llamada masCercano que recibe una lista de números L y un número N, y retorna el elemento mayor de L que no es mayor que N. Por ejemplo:

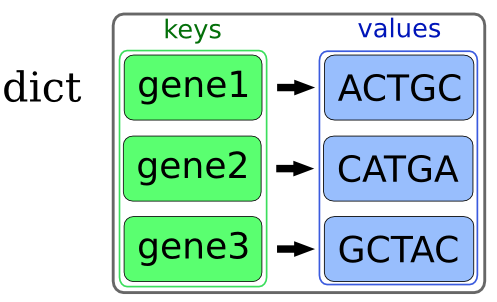
L = [1, 6, 3, 9, 11]

N = 8

La función debería retornar 6. Porque 6 es el número más cercano en L a 8, que no es mayor que 8.

Escriba una función llamada merge, la cual recibe como parámetros dos listas que ya están ordenadas (pueden ser de diferente longitud), y las une en una sola lista ordenada.

1. Realice esto utilizando la función sort.
2. Realice esto sin utilizar la función sort.



Cree una función que reciba como parámetro una cadena de texto donde cada palabra está separada por espacios, y devuelva un diccionario que contenga la frecuencia de las palabras de la cadena. Es decir, cuantas veces aparece cada palabra en la cadena.

Escriba una función que recibe una cadena de texto donde cada palabra está separada por espacios y retorne un diccionario donde las claves serán la longitud de las palabras, y los valores serán las palabras de la cadena que tienen esa longitud.

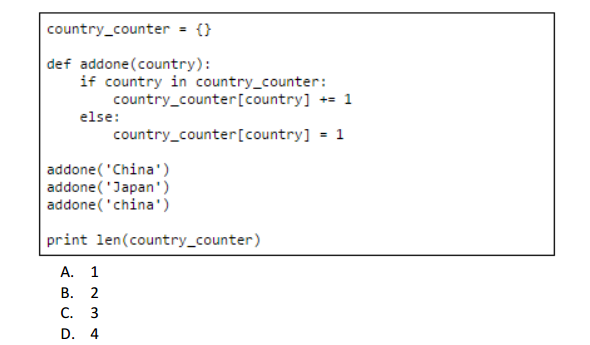
Suponga que usted posee la siguiente lista de strings:

Lista = [‘Arturo Mina|3’, ‘Lionel Messi|10’, ‘Cristiano Ronaldo|7’, ‘Junior Sornoza|10’,

‘Gerard Pique|3’… … … ‘Jugador X|9’]

Cada elemento de la lista tiene los atributos “nombre|#camiseta”

Realice una función que reciba como parámetro una lista como la mostrada anteriormente y devuelva una estructura en donde se tenga guardado todos los nombres de los jugadores agrupados por mismo número de camiseta.



Escribir una función que reciba una cantidad de iteraciones de una tirada de 2 dados de 6 caras al azar y devuelva la cantidad de veces que se observa cada valor de la suma de los dos dados.

Dado un diccionario con la siguiente estructura:

{“@usuario1”:[“contenido #hashtag1”, “contenido #hasthag2 contenido”],

“@usuario2”:[“#hashtag2 contenido #hasthagn mas contenido”],

... }

Elabore una funcion llamada trendTopics(diccionario) que reciba un diccionario con la estructura anterior y retorne un segundo diccionario de hashtags con el numero de veces que fueron utilizados en los tuits. Luego, elabore un programa que muestre por pantalla los hashtags y sus contadores. Asuma que existe la funcion crearDiccionario() que retorna un diccionario con los tweets de acuerdo a la estructura presentada arriba.

Ejemplo de salida:

#hashtag2 2

#hashtag1 1

#hashtagn 1

