Observaciones – Laboratorio 3

a. ¿Cuáles son los mecanismos de interacción (I/O: Input/Output) que tiene el view.py con el usuario?

Se puede cargar información, buscar el top de libros, buscar según autor, buscar según etiqueta. A través del menú de usuario mostrado en la ilustración 1.

```
def printMenu():
    """
    Menu de usuario
    """
    print("Bienvenido")
    print("1- Cargar información en el catálogo")
    print("2- Consultar los Top x libros por promedio")
    print("3- Consultar los libros de un autor")
    print("4- Libros por género")
    print("0- Salir")
```

Ilustración 1 - Menú de usuario

b. ¿Cómo se almacenan los datos de GoodReads en el model.py?

Los datos de GoodReads en el model.py se almacenan en un catálogo el cual se crea mediante la función newCatalog (), esta función crea una lista vacía para los libros, otra para los autores y otra para los tags, esto creado a partir listas que provienen de la librería DISC. En la ilustración 2 se muestra la función newCatalog ()

```python
def newCatalog():
    """
    Inicializa el catálogo de libros. Crea una lista vacia para guardar
    todos los libros, adicionalmente, crea una lista vacia para los autores,
    una lista vacia para los generos y una lista vacia para la asociación
    generos y libros. Retorna el catalogo inicializado.
    """
    catalog = {"books": None, "authors": None, "tags": None, "book_tags": None}

    catalog["books"] = lt.newList("SINGLE_LINKED")
    catalog["authors"] = lt.newList("ARRAY_LIST", cmpfunction=compareauthors)
    catalog["tags"] = lt.newList("ARRAY_LIST", cmpfunction=comparetagnames)
    catalog["book_tags"] = lt.newList("SINGLE_LINKED")

    return catalog
```

Ilustración 2 - Función newCatalog ()

c. ¿Cuáles son las funciones que comunican el view.py y el model.py?

Las funciones que comunican el view.py con el model.py son las siguientes:

- newController (), función encontrada en el view.py que se comunica con la función newController (), encontrada en el archivo controller.py, esta a su vez se comunica con la función newCatalog ().
- La función loadData (), encontrada en el view.py, la cual se comunica con el controlador y se comunica con las funciones loadBooks (), loadTags (), loadBooksTags (), sortBooks (). Las cuales se comunican con el modelo a través de las funciones addBook (), addTag (), addBookTag () y sortBooks () respectivamente.
- Las funciones getBestBooks (), getBooksByAuthor (), countBooksByTag () que se comunican con el controller.py y estas a su vez con el model.py ()

d. ¿Cuál es la función que permite crear una lista? ¿Qué datos son necesarios?

La función que permite crear una lista es la función lt.newlist (). Esta función se puede observar en la ilustración 3. No hay datos que sean necesarios debido a que la función newList () trae valores por default. Sin embargo, si se desean establecer estos datos, existen dos principales, que son: “datastructure” y “cmpfunction”

```
def newList(datastructure='SINGLE_LINKED',
            cmpfunction=None,
            key=None,
            filename=None,
            delimiter=","):
```

Ilustración 3 - Función NewList

e. ¿Para qué sirve el parámetro datastructure en la función newList ()? ¿Cuáles son los posibles valores para este parámetro?

El parámetro datastructure sirve para determinar el tipo de estructura de datos a utilizar para implementar en la lista. Un ejemplo es el encontrado en la ilustración 4.

Los posibles valores para este parámetro pueden ser: ARRAY_LIST, SINGLE_LINKED y DOUBLE_LINKED. Sin embargo, en este laboratorio solo son utilizados los dos primeros valores nombrados.

```
catalog["books"] = lt.newList("SINGLE_LINKED")
```

Ilustración 4 - Ejemplo del tipo SINGLE_LINKED de datastructure

f. ¿Para qué sirve el parámetro cmpfunction en la función newList ()?

Sirve para hacer una comparación entre los elementos de una lista. Esta función decide si dos elementos son iguales o no. Sirve para encontrar si una variable existe en la lista. Como, por ejemplo, comparar los autores de una lista, tal como se encuentra en la ilustración 5.

```
catalog["authors"] = lt.newList("ARRAY_LIST", cmpfunction=compareauthors)
catalog["tags"] = lt.newList("ARRAY_LIST", cmpfunction=comparetagnames)
```

Ilustración 5 – Ejemplo de la función cmpfunction

g. ¿Qué hace la función addLast ()?

La función addLast () agrega un elemento en la última posición de la lista. Al implementarse no solo se agrega el elemento, sino que también se actualiza el apuntador a la última posición y se incrementa el tamaño de la lista en 1.

h. ¿Qué hace la función getElement ()?

La función getElement () recibe un parámetro POS y retorna el elemento en dicha posición POS sin eliminarlo. Es importante tener en cuenta que la posición debe ser mayor que 0 y menor o igual al tamaño de la lista. Además, la lista no puede estar vacía.

i. ¿Qué hace la función subList ()?

Es una función que retorna una sublista de una lista dada por parámetro. La función retorna una lista que contiene los elementos a partir de la posición pos. Crea una copia de dichos elementos y los retorna en una nueva lista.

j. Revise el uso de la función iterator () en las funciones printAuthorData (author) y printBestBooks (books) en la Vista que aplican a una lista de libros. ¿Qué hace la función iterator ()?

La función iterator permite que las funciones del programa puedan iterar sobre las listas. Es decir, que tengan la capacidad de repetirse.

k. ¿Observó algún cambio en el comportamiento del programa al cambiar el valor del parámetro 'datastructure' en la creación de las listas?, explique con sus propias palabras los cambios que notó.

No notamos ningún cambio, esto se debe a que únicamente se cambia el tipo de estructura de datos, pero siguen siendo listas y siguen funcionando como tal. De hecho, al funcionar a través de un TAD y como ambas estructuras tienen las mismas funcionalidades entonces debe seguir funcionando igual. De hecho, muy probablemente tengan tiempos de ejecución diferentes en algunas funcionalidades.