OBSERVACIONES DEL LA PRACTICA

Sebastián Gómez Cod 201912614
Andrés Santiago Martínez Hernández Cod 201921006

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

   El usuario ingresa mediante un input la acción que quiere ejecutar, ya sea cargar datos o realizar acciones sobre los mismos. Después de la ejecución de la función invocada el retorno se muestra al usuario mediante un print (los 5 mejores libros, libros por autor, etc.)

2) ¿Cómo se almacenan los datos de **GoodReads** en el **`model.py`**?

   Los datos son almacenados en un diccionario (catalogo) cuyas llaves corresponden a TADLists (Books, Authors, tags, bookTags); para el caso de authors, tags y bookTags son ArrayLists.

3) ¿Cuáles son las funciones que comunican el el **`view.py`** y el **`model.py`**?

   En de view.py (soportadas por controller):

   initCatalog() <crea el catalogo vacío>
   loadData() <llama a controller para que este, a su vez, llame a model para cargar los datos>
   getBestBooks() <retorna un TADList con el numero de libros indicados según una característica>
   getBooksByAuthor()<retorna un TADList con los libros que comparten el mismo autor>
   countBooksByTags()

4) ¿Cómo se crea una lista?

```
(function) newList: (datastructure='SINGLE_LINKED', cmpfunction=None, key=None,
filename=None, delimiter=",") -> dict[str, Any]
```

   Invocando la función de DISClib llamada newList, esta recibe como parámetros el tipo de estructura de datos, función de comparación, llave de comparación (si no se da la cmpfunction), filename para crear una lista basados directamente en un archivo plano (el delimitador del archivo está incluido como delimiter).

5) ¿Qué hace el parámetro **`cmpfunction=None`** en la función **`newList()`**?

   El valor None establece la función de comparación por defecto como la función para comparar. En este caso, se debe suministrar el parámetro key para identificar sobre qué llave se hará la comparación en la lista.

6) ¿Qué hace la función **`addLast()`**?

Añade en la última posición de la lista (dependiente del tipo de estructura de datos) el elemento dado por parámetro.

7) ¿Qué hace la función **`getElement()`**?

   Retorna el elemento en X posición/índice de la lista.

8) ¿Qué hace la función **`subList()`**?

   Crea una copia de la lista ingresada por parámetro desde la posición ingresada por parámetro y de la longitud ingresada por parámetro.

9) ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro **“ARRAY_LIST”** a **“SINGLE_LINKED”**?

   Sí, aunque la complejidad en ambos casos es O(n), se percibe que con SINGLE_LINKED es un poco más rápido. Sin embargo, a nivel global el programa continuó trabajando normalmente.