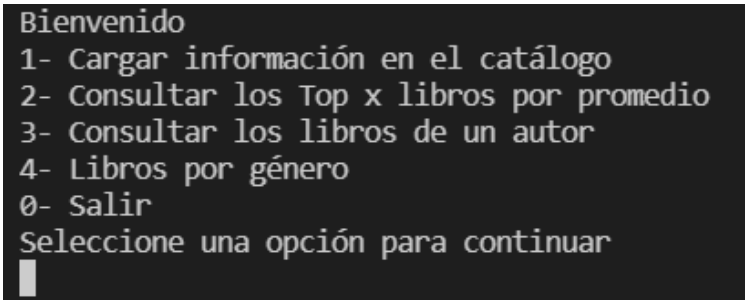
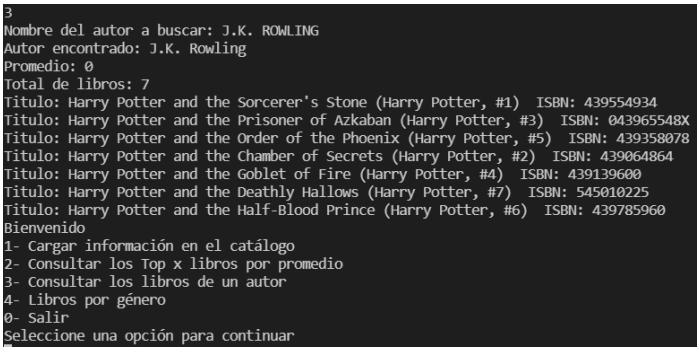
OBSERVACIONES DEL LA PRACTICA

Estudiante 1 Juan Manuel Pérez S Cod 202021827
Estudiante 2 Nicolas Camargo Prieto Cod 202020782

1) ¿Cuáles son los mecanismos de interacción (I/O: Input/Output) que tiene el **`view.py`** con el usuario?
= El view.py se encarga de mostrar un menu con las opciones a elegir por el usuario solicitandole un input, el cual es un numero del 0 al 4 y cada numero tiene una opcion diferente que solicita al controller.py y al model.py que realice ciertas funciones y retorne al view.py dicho resultado de esas funciones, con el fin de que el view.py le muestre esta informacion al usuario de una mejor manera (output).



Ademas, una vez le muestra la informacion solicitada al usuario, el view.py le vuelve a pedir un input para continuar con el programa o bien para salir.



2) ¿Cómo se almacenan los datos de **GoodReads** en el **`model.py`**?
= Al cargarse los datos de los diferentes archivos en el controller.py, el model.py lo que hace es inicializar un catalogo de libros por medio de un arreglo de listas vacias (‘ARRAY_LIST’) en la funcion newCatalog. Por lo cual, se almacenan los datos de libros, autores, tags y tags de libros bajo las referencias: ‘books', 'authors', 'tags' y 'book_tags'.

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

    return catalog
```

Posteriormente, retorna el catalogo creado, es decir este arreglo de listas con las referencias, el cual es usado en el controller.py, para solicitarle al model.py que vaya agregando al arreglo cada dato del archivo .csv, y el model.py lo que hace es ir anadiendo los datos a ese arreglo vacio y guardandolos bajo las referencias mencionadas.

```python
def loadBooks(catalog):
    """
    Carga los libros del archivo.  Por cada libro se toman sus autores y por
    cada uno de ellos, se crea en la lista de autores, a dicho autor y una
    referencia al libro que se esta procesando.
    """
    booksfile = cf.data_dir + 'GoodReads/books.csv'
    input_file = csv.DictReader(open(booksfile, encoding='utf-8'))
    for book in input_file:
        model.addBook(catalog, book)
```

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

= En realidad ninguna directamente, sin embargo por medio del controller.py el view le solicita funciones al model.py, de esta forma siguiendo el patron MVC. De la misma forma, el model.py le manda los datos solicitados al controller y este los pasa al view.py para que sean mostrados en el input al usuario.

```python
47  def initCatalog():
48      """
49      Inicializa el catalogo de libros
50      """
51      return controller.initCatalog()
52
53
54  def loadData(catalog):
55      """
56      Carga los libros en la estructura de datos
57      """
58      controller.loadData(catalog)
```

Por ejemplo, las funciones initCatalog y loadData requieren que el controller interactue con el model.py para que inicialice el catalogo de libros y cargue los libros en un arreglo.

4) ¿Cómo se crea una lista?

= Una lista comun en python se crearia con una variable y asignando el contenido de la lista con corchetes.

Ej:

Lista1 = [‘juan’,’pedro’,’farid’]

```
catalog['books'] = lt.newList()
catalog['authors'] = lt.newList('ARRAY_LIST',
                                cmpfunction=compareauthors)
```

En el laboratorio se crea usando la funcion newList(). En este caso, se crea un arreglo mediante el cual cada lista de informacion (libros, autores, tags, etc.) guardara los datos bajo las referencias: ‘books’, ’authors’, ’tags’, etc. Tambien en este caso se usa una l ibrería llamada ‘DISClib.ADT’ donde se importa la lista bajo el nombre de lt.

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

El parámetro cmpfunction en la función newList es una función de comparación para que los elementos no se repitan en la lista o se puedan ordenar. En el ejemplo de los libros, había un cmpfunction sobre los autores para que estos no se repitieran si tenían más de un libro.

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

La función addLast recibe como parámetro la lista y el elemento que se quiere agregar en la última posición (nodo). De igual forma se actualiza el apuntador a la última posición que es útil para las listas encadenadas y incrementa el tamaño en 1.

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

La función getElement recibe como parámetro la lista y una posición de un elemento. Se recorre la lista hasta llegar a la posición deseada y retorna el elemento en esa posición. En una lista encadenada se recorren todos los elementos mientras que en un arreglo se sabe cual es la posición. La lista no puede estar vacía y la posición debe ser un número mayor a cero o igual al tamaño de la lista.

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

La función subList recibe como parámetro la lista, una posición y un número de elementos que se quieren copiar. A partir de la posición dada en la lista se crea una nueva lista con una longitud de el número de elementos que se querían copiar. Retorna otra lista.

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

El programa no cambio siguió funcionando igual como cuando trabajaba mediante ARRAY_LIST se probaron las 4 opciones en el menú y para ambas implementaciones el resultado fue el mismo para las dos.