**Laboratorio 3**

**Grupo 5**

**Daniel Reales**

**José Vicente Vargas Panesso**

**Paso 3:**

1. ¿Cuáles son los mecanismos de interacción que tiene el view.py con el usuario?

El primer mecanismo presente en el código está en la función “printMenu()” por medio de la cual se imprimen por consola y por lo tanto se le informa al usuario los comandos disponibles para las diferentes funcionalidades del código.

Posteriormente durante la ejecución del menú principal este código presenta un “input” por medio del cual se obtiene la respuesta del usuario del comando que se quiere implementar, y por lo tanto lo que se desea hacer.

Finalmente, al ejecutar alguna de las funcionalidades del código, se llama una función específica, que puede ser “printAuthorData(author)” o “printBestBooks(books)”, las cuales implementas métodos de impresión por consola para comunicarle al usuario lo que necesita según la funcionalidad que deseó implementar.

1. ¿Cómo se almacenan los datos de GoodReads e el model.py?

Inicialmente la función “newCatalog()” presente en el modulo model.py, se encarga de inicializar un diccionario con 4 llaves, una por cada uno de los posibles datos que se pueden obtener de GoodReads, como lo son “books”, “tags”, “books\_tags” y “authors”. Cada una de estas llaves tiene asociada una lista con diferente estructura de datos. Específicamente la llave “books” tiene asociado una estructura de datos tipo “SINGLE\_LINKED” mientras que las demás tienen asociadas una estructura de datos tipo “ARRAY\_LIST”.

Finalmente por medio de las funciones “addBook(catalog, book)”, addBookAuthor(catalog, authorname, book), addTag(catalog, tag) y addBookTag(catalog, booktag), se adicionan los datos del archivo csv dados por parámetro a la lista correspondiente dentro del diccionario catalog.

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

Las funciones que comunican estos dos módulos se encuentran en el controller.py, esto dado que el esquema del código sigue el esquema vista-modelo-controlador.

Específicamente, las funciones initCatalog(), loadData(catalog), getBestBooks(catalog, num), getBooksByAuthor(catalog, authorname) y countBooksByTag(catalog, label), son las que se encuentran en controller.py y que implementan la conexión entre en view.py y model.py. Las dos primeras funciones se encargan de inicializar y cargar los datos del archivo csv, de la forma ya especificada en la pregunta anterior. Mientras que las otras tres funciones se encargan de implementar funcionalidades especificas del código, como entregar los mejores libros, encontrar los libros escritos por determinado autor, y encontrar los libros por género, respectivamente.

**Paso 4:**

1. ¿Cómo se crea una lista?

El archivo list.py para poder crear una lista nueva tiene una función llamada newList(dataestructure, cmpfunction, key, filename, delimiter), la cual recibe necesariamente como parámetro el tipo de estructura de datos deseada, que puede ser “ARRAY\_LIST” o “SINGLE\_LINKED”. También puede recibir una función que define como se puede hacer la comparación entre dos elementos, la llave que es necesaria para hacer la comparación (de ser necesario), el nombre de un archivo de donde se quiera cargar la información (de ser necesario), y el tipo de carácter con la cual se realiza la separación de datos en el archivo del cual se quiera cargar la información.

Es necesario agregar que para poder crear la lista del tipo de datos especificado, este archivo llama al modulo “liststructure.py”, el cual a su vez llama a los archivos “arraylist.py” y “singlelinkedlist.py”, que contienen los métodos necesarios para crear una estructura de datos tipo array y tipo lista enlazada, respectivamente.

1. ¿Qué hace el parámetro cmpfunction=None en la función newList()?

Este parámetro define la función que se quiere usar al momento de comparar dos elementos de una lista, la notación según la cual está escrita la función indica que, de no definir una función comparados, no es necesario ingresarla por parámetro, ya que la función le da un valor de None, si no es ingresado ningún valor por parámetro.

Es importante resaltar que al dejar el valor None por defecto, los archivos “arraylist.py” y “singlelinkedlist.py” dentro de sus funciones “newList()”, establecen que en dado caso que este parámetro sea None, se asigna la función comparadora por defecto creada en estos dos archivos.

1. ¿Qué hace la función addLast()?

Esta función añade un nuevo elemento al final de la lista sea cual sea el tipo de esta. Para esto recibe como parámetro la lista a la cual se añadirá el nuevo elemento, y el nuevo elemento que se quiera agregar.

1. ¿Qué hace la función getElement()?

Esta función retorna el elemento en una posición específica, para esto recibe como parámetro la lista y la posición de la cual se desea obtener el elemento.

1. ¿Qué hace la función subList()?

Esta función crea una sub lista basada en los elementos de una lista original. Para esto recibe como parámetro la lista original de la cual se obtendrán los elementos, la posición desde la cual se creará la sub lista, y el número de elementos que esta tendrá. Es decir se añadirán elementos a la sub lista desde la posición especificada, de forma consecutiva según la cantidad de elementos especificada.

**Parte 5**

Para verificar el tiempo de ejecución en la carga de los datos se creó un indicar de tiempo por medio de la librería time, tal y como se muestra en el fragmento de código posterior.



1. En la primera prueba de cargar los datos, al modificar la estructura datos por “SINGLE\_LINKED”, se encontró un tiempo de carga final de 6 minutos y 44 segundos. Al comparar este tiempo con el obtenido al trabajar con la estructura de datos “ARRAY\_LIST” (5 minutos 46 segundos), se puede evidenciar un aumento en el tiempo, esto se debe a que en la creación estructura “SINGLE\_LINKED” se debe guardar tanto la información como el apuntador.
2. En la prueba de mostrar los mejores libros se generó un tiempo de 1958700ns, que al compararlo con el tiempo de ejecución con “ARRAY\_LIST” de 996400ns, resulta ser aproximadamente el doble. Esto se explicaría dado que la función “getBestBooks()” con la cual se obtiene la información de los mejores libros, dentro del archivo “model.py” utiliza la función “getElement()” la cual se sabe que tiene una complejidad O(n) para el tipo “SINGLE\_LINKED”, mientras que para el “ARRAY\_LIST” tiene una complejidad de O(1), lo cual explica la diferencia de tiempos en el algoritmo según el tipo de datos implementado.
3. En la prueba de libros por autor el programa se demora un tiempo de 6980100ns, lo cual es aproximadamente igual al tiempo que toma e la ejecución inicial. Esto se debe a que la función “getBooksByAuthors()” implementa un recorrido sobre la lista que se encuentre en catalog[“authors”], la cual no se pide modificar durante el desarrollo de este laboratorio, es decir que esta implementación siempre se está realizando con una estructura de datos tipo “ARRAY\_lIST”.

Por otro lado, se sabe que si se cambiara la estructura de datos de esta lista por una de tipo “SINGLE\_LINKED”, se vería un incremento en el tiempo de ejecución de la función. Esto por el mismo motivo que en el numeral anterior, dado que el procedimiento implementa la función “getElement()” que como se explicó anteriormente presenta un crecimiento temporal mucho mayor para una estructura tipo “SINGLE\_LINKED” en comparación con una estructura tipo “ARRAY\_LIST”.