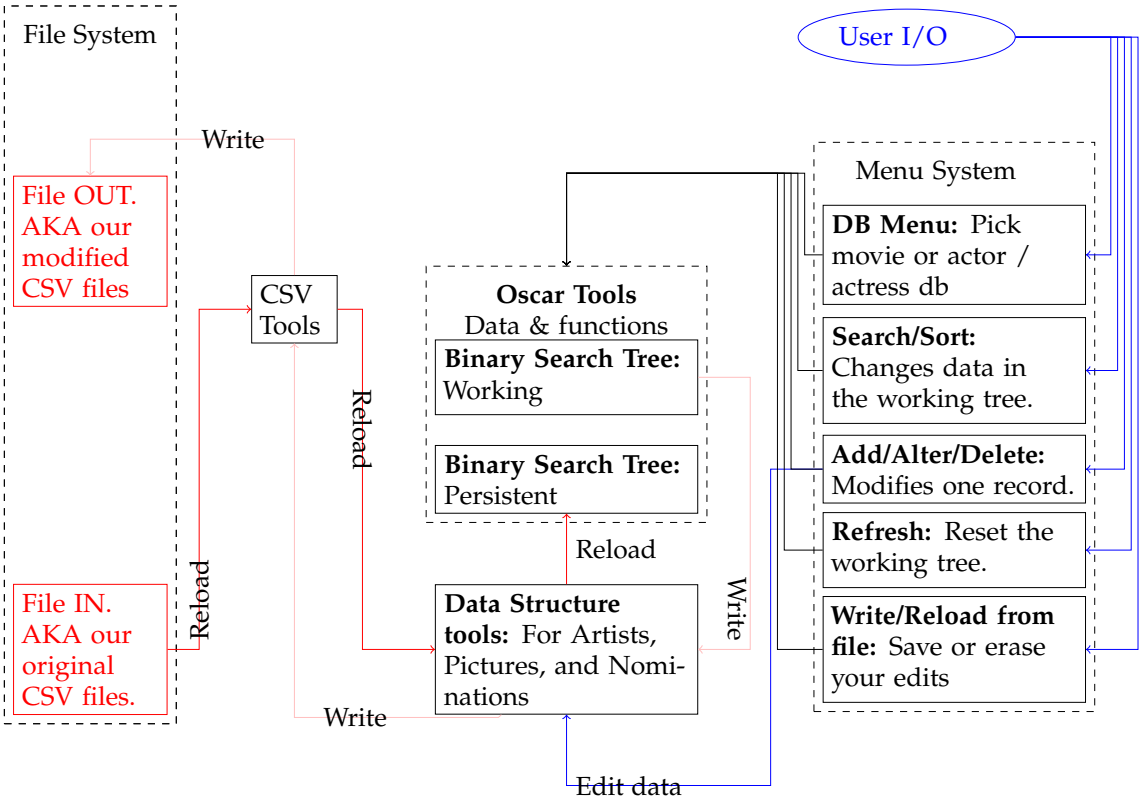
OSCAR DATABASE, WITH COMMAND LINE MANAGEMENT

RORY FLYNN (11/1/2018)

1. Problem Description

This program provides a simple and intuitive interface to manage and edit large amounts of data related to the Oscars. It includes a wide array of functions to organize and alter the data via command line. You can read in comma(",") separated files of data related to movie pictures, actors and actresses, or nominations.

2. Overall Software Architecture



Note: *In this picture the Blue lines represent direct user input.*

2.1. **Menu System.** The menu system is where the user interacts with the program. However it's largely a collection of formatted prompts. Anything meaningful is handled by the Oscar tools system or the data structures. There are two important menus for the program. The first is the database menu, which decides what data template will be given to the Oscar tools. Note that this will determine what database can be used. The second is the action menu, where you select the action to take on the data in that system. The action menu also allows you to navigate through a

Date: 11/1/2018.

large selection of data. Many items in the action menu may have sub-menus that use the tools of the action menu. You can exit the program at any time from any menu by typing q or Q.

2.2. **Oscar Tools.** When an action is taken in the menu system, the user's choice this passed to the Oscar tools system. This portion of the program will either act on the user's input immediately or pass it to the appropriate data structure. This is by far the most complex system in the program and has two parts, the Oscar Pointer, and the Oscar Database. The Oscar Database inherits from the Oscar Pointer, which is an abstract class. Doing this allows for a pointer to switch between templated database structures without any complications. The most important components of the Oscar DB class are the binary search trees of which there are two. The working binary search tree holds data that is currently being worked on or modified, and is affected by all actions. The persistent tree however is only affected by delete, add, or modify actions. The most important functions in the class manage these two trees.

2.3. **Data Structure Tools.** At first glance the data structure classes might seem like the least important aspects of the program. However these files control how data is read into the program and how it is used in the system. These classes contain the data types for all variables related to motion pictures, actors and actresses, and nominations respectively. They also determine how the data is to be displayed, and even how the data can be manipulated. They are the most custom, and therefore vital, classes in the system. The most important functions in these data structures are the functions that relate to acceptable input/output from the user.

3. Input Requirements

The program accept CSV files exclusively. The CSV file must be located in the working directory of the program. The fields of the CSV must be in the order depicted in the charts below, and must be of the data type described or a No-Entry flag will be assigned to the input. For all data types except nominations the name field is required, and the program will halt if it is not provided. The keyboard input holds the exact same requirements although you can use a menu system to enter values in any order. If the user fails to give a name for any data type they will be re-prompted to try again.

Pictures CSV Format.

CSV Column	Data Type	Name	Notes
0	string	name	Can not be blank
1	Int	year	
2	Int	nominations	
3	Double	rating	
4	Int	duration	
5	string	genre1	
6	string	genre2	
7	string	release	
8	Int	metacriticticEmty	
9	string	synopsis	

Actor/Actress CSV Format.

CSV Column	Data Type	Name	Notes
0	Int	year	
1	string	award	
2	Bool	winner	$1 = true, 0 = false$
3	string	name	Can not be blank
4	string	film	

Nominations CSV Format.

CSV Column	Data Type	Name	Notes
0	Int	year	
1	string	award	
2	Bool	winner	$1 = true, 0 = false$
3	string	name	Ok to be blank
4	string	film	

4. Output Requirements

This program is designed to write user feedback to a standard Linux command prompt using ASCII encoding. When the option is selected by the user, it will output a CSV file to the working directory. The output CSV file is formatted the same as the input above.

5. Problem Solution Discussion

Ultimately this program required very few complex calculations. The most complex structure in the entire system is the binary search tree.The BS tree is detailed in the data structure section of this document. No algorithm in the entire program in fact uses a method for the binary search tree that runs in more than O(n) time. Furthermore, no program in this contains more than a single loop. So it's safe to say that no action in this system will take more than O(n) time.

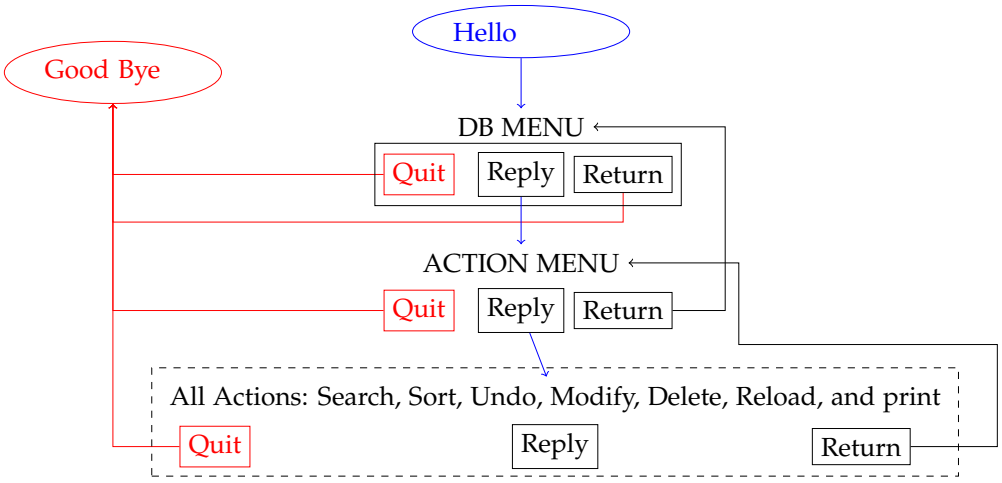
6. Data Structures

I opted for a set of two Binary search Trees, and a vector for data that the user manually selects. Even if it was not a requirement I would likely still have been drawn to the BS Tree for this project.

For my design I need to (A) be able to loop through the data quickly and simply, and (B) be able to find/sorted data items quickly. A hash table would excel at (B) but not (A). A linked list, or vector, would work for (A) but could not easily achieve (B). A BS tree can mimic the linear functionality of an array but it is much faster to search when using a key. This makes it perfect for my needs.

7. User Interface Scheme

The menu structure of this program is very simple. The menu uses one universal prompt. The user can enter an R to return to the menu above (exit if none), a Q to exit the program, or a number representing a valid choice. No other input is accepted. When the user first enters the program, they are greeted with a database menu where they can pick which database they would like to load. Then they are taken to the action menu where they can select the action they wish to take, or view more data. When a user selects an action, they are dropped into the action sub program. All sub programs contain the option to return to the action menu. Note that pressing q in any menu exits the entire program.



8. Status of Application

To the best of my ability to determine program implements all the required functionality with some detectable bugs. I believe that if I understand the assignment correctly then nothing in this program should be found lacking. This is a large program but I did my best to check it thoroughly.

Bugs. – I have observed the following bugs, but I don't have time to fix them.

You can delete any node in the database except the root node. The the program crashes.

If you load all databases at the same time you will end up with six trees total. Usually this is fine on the server, but during nights of heavy traffic like this it seems to slow the tree down a lot.

CSE grid. The program has been compiled and tested on the CSE grid with all data bases loaded including nominations.

Extra Credit Status. The nominations database has been created and functions well to the best of my knowledge.

Statistics on nominations have not yet been implemented.