*Namito Yokota*

*UNIVERSITY OF ALABAMA | MARCH 29TH, 2019*

Explaining the Code

IMDB Catalog

**FILE STRUCTURE**

**imdb-catalog/**

**|\_\_\_\_\_\_ bin/**

**|\_\_\_\_\_\_ main**

**|\_\_\_\_\_\_ data/**

**|\_\_\_\_\_\_ movie\_records.tsv**

**|\_\_\_\_\_\_ doc/**

**|\_\_\_\_\_\_ \*.docx**

**|\_\_\_\_\_\_ lib/**

**|\_\_\_\_\_\_ \*.h**

**|\_\_\_\_\_\_ log/**

**|\_\_\_\_\_\_ \*.log**

**|\_\_\_\_\_\_ src/**

**|\_\_\_\_\_\_ \*.c**

**|\_\_\_\_\_\_ README.md**

**|\_\_\_\_\_\_ makefile**

**|\_\_\_\_\_\_ .gitignore**

**MAIN**

The main.c file in this project simply works as a front-end using ncurses. With back-end connected through this file, users can interact with the program visually.

Menus

To create different menu windows, I create a function for each window that needs to be created. When the menu needs to be displayed, the main function can easily call the specific menu function.

Parse

At the time of execution, a critical part of this program is to parse the download IMDB database to store in the memory. The function parseFILE() works as the name suggests. It first starts off by checking that the file is downloaded, then it parses the file line by line to create a new red black tree.

Global Variables

To simplify the process of passing in variable to each function with required information, many variables that are repeatedly used throughout the project are stored in globally.

**RED BLACK TREE**

Red black tree is my data structure of choice for storing movie datasets. Using index as the value to sort movies, red black tree allows most operations in time.

Insert and Delete

Fixups

After each insertion or deletion of a node, ofter, properties of red black trees are broken. To fix this problem, functions like \_\_\_\_ are used.

Extras

Tree walks, search,

**STACK**

Red black tree works as a primary data structure for this project. Although, to print out each movie in order to the user, using simple tree walk algorithms can be difficult especially with ncurse UI. To assist with this problem, I used a stack data structure.

By operating a inorder tree walk which pushes each node to the stack, the main file can print the list in the stack by simply popping each movie from the list.

Push and Pop

**CRUD**

CRUD (Create, Retrieve, Update, and Delete) can be argued as one of the main features in this program as well. The idea is that users can edit a record from the red black tree at any point. To allow this process, I created a CRUD file to operate each of the functionalities.

Create

This function simply inserts a new node to the existing red black tree

Retrieve

This function searches for a movie that has been previously deleted, then restore that movie by updating its information.

Update

This function updates a record of their choice by assigning new literals to the existing variables.

Delete

This function works like a retrieve function. It updates the record but this time to make it invisible for the users.

**LOGGER**

Another factor that differentiates this project from a simple data structure project is that the program keeps logs of all of your operations. Inside each of the CRUD operations, the function writeLog() is called to keep history. This allows you to load history when you log back in next time.

printLog()

The function takes in a parameter of the message that needs to be written as well as the name of the username. The username is used to create a new file then the message is written followed by a new line.

readLog()

This function goes along with the print because when the user logs back into the program, the main function automatically calls readLog to see their previous operations. Then, all of the operations are completed to keep them up to date.