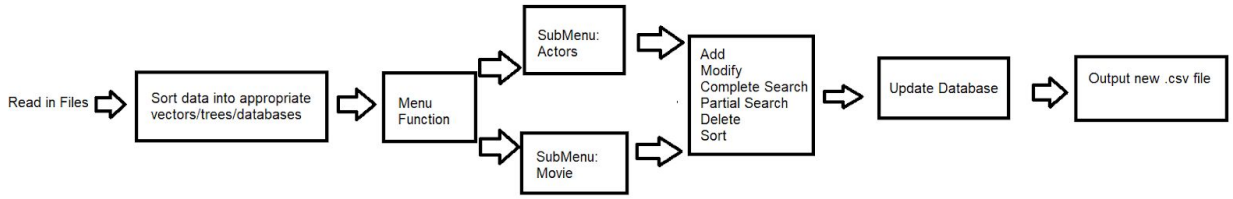
Project Design/Implementation Document
1. Title Page

Title: Database of Academy Award Winners
Team Name: Simmering Yellow Groundhogs
Team Members: Kerry Gip & Odyssey Villagomez
Due: Nov 8th 2020

2. Problem Description
The program reads in a file and stores multiple different kinds of data and stores them in different vectors and databases
From the vectors, the user can search for different features, sort records and produce outputs. The user can also add and delete records and modify them.

3. Overall Software Architecture
A brief description of major functions and their main roles in the program. You need to explain how the entire program is constructed and how the functions are related to each other. You don't have to explain every little function. A diagram to display relation is very useful to get an overall picture. Here's an example of the diagram for a different program but you'll get an idea.



***See attached png file for full size**
The program will read the data of actors or movies into a binary search tree or vector that will display different tables containing rows and columns. A menu function will then direct which tree the user wants to search/modify the data. After the modification, the database is updated and the user has the option to print out the updated data.

4. Input Requirements
A detailed list of all external inputs (from files or keyboard) including a description of the data type and range of valid values for each input. For input file format and interactive user input, you need to write what data type is used for every field and valid value and length.

Two .csv files:

- actor-actress.csv

Must contain all appropriate data (Year, Award, Winner, Name, Film)

- pictures.csv

Must contain all appropriate data (Name, Year, Nomination, Rating, Duration, Genre1, Genre2, Release, Metacritic, Synopsis)

User prompts:

- **User can choose either actor or movie database and add a record**
- **User can choose either actor or movie database and modify searched fields**
- **User can choose either actor or movie database and sort any field**
- **User can choose either actor or movie database and do a complete search and have an exact match**
- **User choose either actor or movie database and do a partial search on any field**
- **User can choose either actor or movie database and print out a .csv file of latest database after their modifications of any above.**

5. Output Requirements
A detailed list or description of all outputs (to files) including a description of the data type and range of valid values for each output.

Screen outputs:

- **A Main Menu function where the user can look up specific data in**
- **A subMenu function that the Main Menu took the user to**
- **Search functions that allow the user to choose between "exact" or "contains"**
- **A secondary Search function that allows the user to start all over again or search within their first query**
- **User prompts**
    - **i.e "Enter the field you want to search, your options are: "**
    - **"Invalid entry, try again"**
    - **"What do you want to do to the entry? Enter "A" to add, "D" to delete or "M" to modify**
    - **"Would you like to search again? If yes, enter "S" if not then what would you like to do? Enter "A" to add, "D" to delete or "M" to modify"**

File outputs:

- **A .csv file of the latest database(after the addition, deletion or modification).**
  - **The .csv file will have its proper headers in the first line**
  - **The .csv file will have the rest of its' data in the subsequent lines**

6. Problem Solution Discussion

A summary description of the solution steps with algorithms analysis (1 paragraph, approximately 100 words). If any unusual techniques or algorithms are used that need further explanation, an additional paragraph may be used.

The first step is to read in the files, that will either be a user prompt or automatically done. As the files are being read, they will be separated into appropriate vectors and binary search trees containing it's specific data (ie. Year, Name, Award, Nomination, Genre etc,). From there, the screen will show a menu that will display the information on screen. The menu will either have switch functions to direct the next steps. The search function for a BST is O(log(n)), traversal is O(n) and in general, BSTs have an algorithmic analysis of $O(\log_2 N)$. The User will then have the option to go into subMenus, with a prompt to go into Movies or Actors. In either subMenu, the user will have another prompt to be able to manipulate the data and output the new data. A lazy deletion takes O(log(n)) and wouldn't change the structure since it's only marked for deletion and not actually deleted. After the user makes his/her changes, the user will have another prompt if they want to either display their changes on screen or print it to an outfile.

7. Data Structures

A description of choice of your data structures and justification. Include a brief explanation for your choice. For example, "I have considered DS1, DS2, and DS3. Their pros and cons are summarized as follow... I choose DS1 over DS2 and DS3 because …."

The main data structure will be a Binary Search Trees(BST) and vectors in it's nodes to store search results.
A BST and vector would be the most efficient way to write the code rather than using a LinkedList, Stack, Queue, Array or Heap.
A LinkedList would not be as efficient in terms of traversing a tree since it only goes one way, a BST allows for traversal to different nodes.
Stacks and queues are not as efficient in terms of adding or removing from the center of the stack/queue. They work more effectively when used with either the addition/removal of the beginning/end of the stack/queue.
Arrays are not dynamic and cannot be used if there are multiple additions to the data, since arrays cannot be implemented with unknown size.

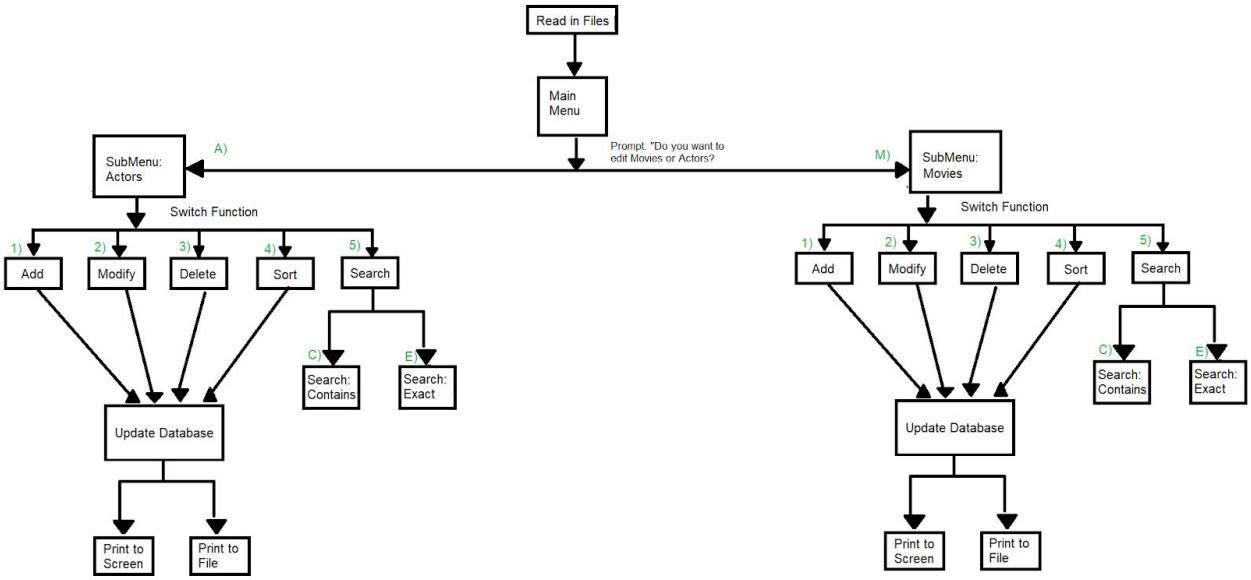
A Heap would not be as effective for the similar reason to a stack/queue. While it may be easy to insert data at the end and remove up front, it does not provide an effective way to insert in the middle of the data.
Vectors would be the easiest to use and manipulate since you can change data positionally and can use a built in sort algorithm.

8. User Interface Scheme
User interface scheme should show the menu items at top level and items in sub menus and how to navigate through menus.

Menu:
Implemented with Switch statement
Select Actors subMenu or Movies subMenu (1 or 2 respectively)

subMenu: Prompt a query type (ie. (1)Add, (2)Delete, (3)Modify, (4)Sort, (5)Search)

Then show a table of results in vector form



9. Status of Application

Note what IDE you developed the Final Project, and whether it compiled and operated properly on the csegrid. Note any requirements that were not met (per the project description). Note any known bugs or issues. If you did extra credit, note the status and what kinds of reports you provided. Include this in the final project submission only.

The IDEs used will be Visual Studio Community Version 2019(Kerry), CLion Version 2020.2.1 (Odyssey) and repl.it