**Project Design/Implementation Document**

1. Title Page Program title, team name, list of team members, and due date?

Title: Movie Buffts

Team Name: Movie Buffs

Team Members: Bryan Heckman, Brian Tillery, Hodan Yusuf

Due Date: November 8, 2020

1. Problem Description A brief description of what this program does?

* This program will develop a simple database system and the database will handle multiple records about movies, each composed of several fields. The database will store its information in a file, it will update the file based on addition & deletion of records and field modifications. It will allow users to sort records based on the selected keys and produce output according to predefined criteria.

1. 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 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?

R1/R2 – Read in data from a text file, .csv, The first two options on the text based menu will be to read in two separate text files into two separate ADT vectors/trees, one for movies and one for actor. These functions will use get line to create a temp variable of our Class Actor or Class Movie and then push them onto the vector<Movie>, vector<Actor> or trees. It will then display success or some type of error and then return the user to the menu.

R3/R4 – Will be to add a record to either database vector or tree. It will call a function which outputs the field names and prompts the user to type in a name using getline when needed. Then create a new variable and push it onto the database passed by reference so that the function can be void but it will modify a vector passed to it by reference.

R5/6 – Will be menu options that ask a user to specify a database, then print out the database and then the user will specify a record to modify and then ask the user for the field values and then replace that element in the database with the new values.

R7/R8 This is a requirement to sort, so it will ask which database to specify and then list the fields that are sortable and then sort the field by that field. I think this will be a function template for sort as it will be passed different values like year or rating and then it will modify the existing database to have sorted values instead of writing a sort function for each field.

R9/R10 Complete search, this will ask the user to specify a database and field and then ask for a complete field and then search the database for that field it will be quicker than a partial search and will use equivalency operator.

R11/R12 Partial search will be more problematic so if were searching for years of a film and we type in 20 we will need to return everything from 2000-present and from 1920 and 1920 or year 200 if that was possible. It will use the search term as a substring of the actual field values.

R13/14 Print database will be similar because it is similar to when you modify a record because it will need to print out before selecting a record to modify. The first line needs to say the column names and the output will have commas separating the values and it should be saved to a fine and printed to the consol.

1. 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?

* We going to have two input files pictures.cvs and nomination.cvs and in the main.ccp we are going to printout our inputs like this by adding nodes.

cout << "Adding Nodes...\n";

tree->addNode(10);

tree->addNode(5);

All the fields will be inputs which searching by that field or modifying or adding a record.

nameOfField, dataType, InputMethod

Year – int , use cin

Award – string, use getline()

Winner – bool, cin

Name --- String, getline()

File – string, getline()

Nomination – int, cin

Rating – double, cin

Genre1 – string – cin

Genre2 – string – cin

Release – string – cin

Metacritic – int – cin

Synopsis – string – getline()

We can use cin for things like complete search for a year field but we will have to use getline for things like searching for titles or names as they have spaces, we will use cin.ignore() inbetween getline and cin.

1. 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?

So the requirements R5/R6/R13/R14 will print out the entire database using a print function to the console in a table format with spaces for easy reading on R5/R6 and with commas separating the fields for R13/R14.

1. 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, and additional paragraph may be used.?

* I think we going to use recursive algorithm and this print using an in order, Depth-first search but we may want to do something else when visiting the node, like moving visited data to another data structure and we will not forget our base case. I know we have to do more than that when it comes solution but for now that is what I can think for. Else if we use a vector will use function to print menu console inputs to select menu options and have modular functions for each option that will be called using a switch statement.

1. 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 .…"

We will be using a vector because we are more familiar with its functions and using it with getline and all the functions of the menu. We might use tree but its really looking like vector is what we want to do.

1. 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.

We will use a do while loop that calls a function that will print out a menu

1. Read in movie database
2. Read in actor database
3. Access movie database
4. Access actor database
5. Quit

Then if the user selects 3 or 4 it will have options for modify, search, print, add, delete, save to file, output to console and back/quit.

While the menu option is not back/quit the do while loop will continue showing the menu.

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.

For many movie lovers, actors and directors, the annual Academy Awards are the highlight of the year. Everyone dresses up, they walk the red-carpet, listen to long and boring speeches and generally pat themselves on the back…but have you ever wondered which movies are the top movies, or who has received the most awards. Well you could ask google, but we are going to do our own data analysis.

The purpose of this final project is to help you think about the design of a somewhat complicated project, then implement and test your code. We want you to start first with the design. Read these requirements and make a design document (answering the questions provided). Create a design document complete with the objects and flow of data, as well as a decision on the best data structures to use for each component. Then you will be ready to implement the code and test. Please don't leave things until the last two weeks. Get started now, and please ask your instructor and TA for help BEFORE you get too lost. Get the big picture done first. Worry about the structure and implementation of the major functionality. Then if you have time, work on the little details, and minor error checking.

So now onto the requirements. ….And the Oscar goes to….

For your Final Project you will develop a simple database system. The database is to handle multiple records, each composed of several fields. The database will store its information in a file, it will update the file based on addition & deletion of records and field modifications. It will allow users to sort records based on the selected keys and produce reports (output) according to predefined criteria.

Some definitions:

1. A database is a collection of information, or data, that you can organize, update, sort, search through, and print as needed. A database does not just hold information; you use a database to organize and analyze information so that you understand its significance.
2. A database file consists of one or more records. Each record holds all the information about one subject item. In C++, the class data type provides an efficient way to represent and manipulate records.
3. Each piece of information in a record is called a field. Fields store the data that has been entered or calculated. In C++, fields are nothing more than the member variables defined for a particular class.
4. **Requirements**
5. Given the requirements as a rough specification, you are to design the classes and implement the database. So you can consider the requirements below as an outcome from a meeting with a client. You are in full control of the choice of data structures, algorithms, and detailed user interface scheme.
6. Requirements are listed with R:
7. You are designing and implementing a database for the Academy Award winners.
8. R1/R2: You are to read in information from two files.
9. R1: You will read in actor-actress.csv which is formatted as a csv file. Place the items in a data structure of your choice, sorted by name.
10. This is a common format and is comma separated (instead of being on separate lines). So you will have comma's between the values. Blank values will just have a comma noting to go to the next field. (so you may have value,, indicating a blank field.) Each line ends in a newline, not a comma. If you want to view the file, often this will be opened by a spreadsheet unless you specifically open it with a text editor. Do not open it with Microsoft Word, as this may change the format. Consider using getline with three parameters, as an easy way to read in files The first line of a CSV file notes the data descriptions as follows:
11. Year,Award,Winner,Name,Film
12. The Winner field has a one if they won and a zero if they did not win.
13. R2: You will read in pictures.csv which is formatted as a csv file. Place the items in a data structure of your choice, sorted by name
14. The first line of pictures.csv contains the data fields including:
15. name,year,nominations,rating,duration,genre1,genre2,release,metacritic,synopsis
16. R3/R4: Choose either the movie or actor database and add a record
17. R5/R6: Choose either the movie or actor database, search for a record, and modify the fields.
18. R7/R8: Choose either the movie or actor database and sort by any single (sortable) field
19. R9/R10: Choose either the movie or the actor database and do a complete search on any “complete” searchable field. It is unlikely that you would have an exact match on an entire description, so that would not be listed for search.
20. R11/R12: Choose either the movie or the actor database and do a partial search on any searchable field. A partial search is any substring within a field.
21. R13/R14: Choose either the movie or actor database, and print out a .csv file of the latest database (after adds, deletes or modifies). Remember that the first line of a .csv file lists the name of the fields separated with commas, ending in a newline. Then the following lines are the information from the fields separated with commas, ending in a newline.

**Database overall management**

1. Use a text based menu for users to choose available features. Program should have a main menu at the beginning and sub menus depending on the task.
2. Each component of the overall program should be fairly modular.
   1. Each menu item, for example, should be a separate function. The menu feature should be handled by a separate function and not by main( ).
3. Program should be fairly fault tolerant of user input (both when the user is entering data, and making menu choices). Provide appropriate user prompts and on-screen directions
4. Split the program into multiple files based on the roughly categorized functionality.

**Data Retrieval and Modification**

1. Users should be able to search records based on the field information in two modes: exact and contains. For example, search "Justin". Then under the search sub menu, users have to pick the search mode (exact or contains) and the field. (Fields should be listed out in another menu, so the user doesn't need to remember a specific field name.
2. Quite often, searches may generate a relatively big output. Users should be able to search again within the search result (secondary search) or start all over again from scratch (new search).
3. Since the entire data is sorted by names, any search (except name of movie or actor) will have to traverse the entire collection and search through the designated fields in every node of the collection.
4. There should be no restriction to the number of records in the database. So, in other words, you should not consider a fixed array for the record data structure.

**Submission Guideline and Due Dates**

|  |  |  |
| --- | --- | --- |
| **Due Date** | **Deliverables** | **Points** |
| 10/25/2020 @ 11:55 PM | Team Information | 100 |
| 11/08/2020 @ 11:55 PM | Design Document | 400 |
| 12/05/2020 @ 11:55 PM | Final Project | 1000 |

**Team Information**

Include team name and list of team members (upto two members per team).

**Design Document**

Design that shows the overall program structures, and the explanation of key algorithms. A description of user interface scheme is required to explain the menu items at top level and items in sub menus and how to navigate through menus. A detailed instruction and sample skeleton is available on Canvas.

**Final Project Submission**

You need to submit following items (all zipped together):

1. Source code with reasonable comments
2. Makefile that works (and is tested) on the csegrid.
3. Readme.txt file noting status of what works and what doesn’t
4. A single final report that includes:
   1. Summary of provided functions. This should be matched with the requirements
   2. Design that shows the overall program structures, and the explanation of key algorithms. A description of user interface scheme is required to explain the menu items at top level and items in sub menus and how to navigate through menus. This is same as the design document stated above.
   3. Accurate status of the program, what's done, and what's not completely implemented.
   4. Accurate status of testing on the csegrid.
   5. The final report should be in MS Word or PDF format.

**Grading Criteria**

1. All work MUST be your own. Any and all help received from another person (other than your team member, TA or instructor), or any source (other than the textbook) must be documented in comments. We reserve the right to give you a zero for the final project if we feel you did not do your own work.
2. Submitting a working program that provides all of the required features will result in a maximum grade of 900 points.
3. Documentation explained above will result in the additional 100 points.
4. Any or all of the following will result in point deductions of up to 5% for each infraction.
   1. Poor and/or inconsistent programming style. This includes the following:
      1. Improper use of indentation.
      2. Overuse of global variables.
      3. Failure to keep functions modular and reusable (possibly applicable to other programs).
      4. Insufficient comments.
   2. Insufficient menu prompts
   3. Program is not reasonably (not absolutely) fault tolerant.
      1. Test to ensure that your program cannot be crashed or sent into an infinite loop by a user who is not following directions.
   4. Partial credit may be awarded.
      1. You may get partial credit for non-working modules (functions) by explaining (in the final document under status) where you think the problem lies.
   5. Submitting a program that does not compile on csegrid.ucdenver.pvt may result in a deduction of at least 20%. Additional points will be lost for each required feature that is not adequately addressed or if we can not easily determine whether your code is correct. Usually if it doesn’t compile you are looking at a 50% penalty since it is difficult to see functionality if it isn’t running.