CSCI 2270 – Data Structures
Summer 2019
Assignment 4
Due Sunday, July 8 by 11:59pm

Binary Search Trees and Linked Lists

An online movie service needs help keeping track of their stock. You should help them by developing a program that stores the movies in a Binary Search Tree (BST) ordered by the first letter in the movie title, and then a singly linked list for each node in the BST that includes the information for the movie. For each of the movies in the store's inventory, the following information is kept:

- IMDB ranking
- Title
- Year released
- Quantity in stock

Your program will have a menu similar to the previous assignment from which the user could select options. In this assignment, your menu needs to include options for finding a movie, renting a movie, printing the inventory, deleting a movie, and quitting the program.

Your program needs to incorporate the following functionality. *These are **not** menu options, see Appendix A for the specific menu options.*

Insert all the movies in the tree.

When the user starts the program they will pass it the name of the text file that contains all movie information. Each line in the file shows the IMDB ranking, title, year released, and quantity in stock. Your program needs to handle that command line argument, open the file, and read all movie data in the file.

From this data, build the BST ordered by the first letter in the movie title. For each of the nodes in the BST, there should be a sorted singly linked list of the actual movie data. *Note: the nodes should be added to the BST and Linked Lists in the order they are read in*. The name of the file that contains the movie data is *Movies.txt*.

Find a movie.

When the user selects this option from the menu, they should be prompted for the name of the movie. Your program should then search the tree and singly linked lists and display all information for that movie. If the movie is not found, your program should display, "Movie not found."

Rent a movie.

When the user selects this option from the menu, they should be prompted for the name of the movie. If the movie is found in your data structure, your program should update the Quantity in stock property of the movie and display the new information about the movie.

If the movie is not found, your program should display, “Movie not found.” When the quantity reaches 0, the movie should be deleted from the singly linked list. If that was the only node in the singly linked list, the node should also be deleted from the BST for that letter.

Print the entire inventory.

When the user selects this option from the menu, your program should display all movie titles and the quantity available in sorted order by title. Refer to the lecture notes as well as the text book on in-order tree traversal, and linked list traversals, for more information.

Delete a movie.

When the user selects this option, they should be prompted for the title of the movie to delete. Your code should then search the tree for the first letter of that movie, and then search the singly linked list for the title. If the title is found, delete it from the singly linked list. If it was the only title for that letter in the BST, you also need to delete the node in the BST and re-assign the parent and child pointers to bypass the deleted node, and free the memory assigned to the node. If the movie is not found in the search process, print “Movie not found” and do not attempt to delete. A movie node should also be deleted when its quantity goes to 0.

Count movies in the tree.

When the user selects this option, your program should traverse the tree and singly linked lists and count the total movie nodes in the tree and print the count.

Quit the program.

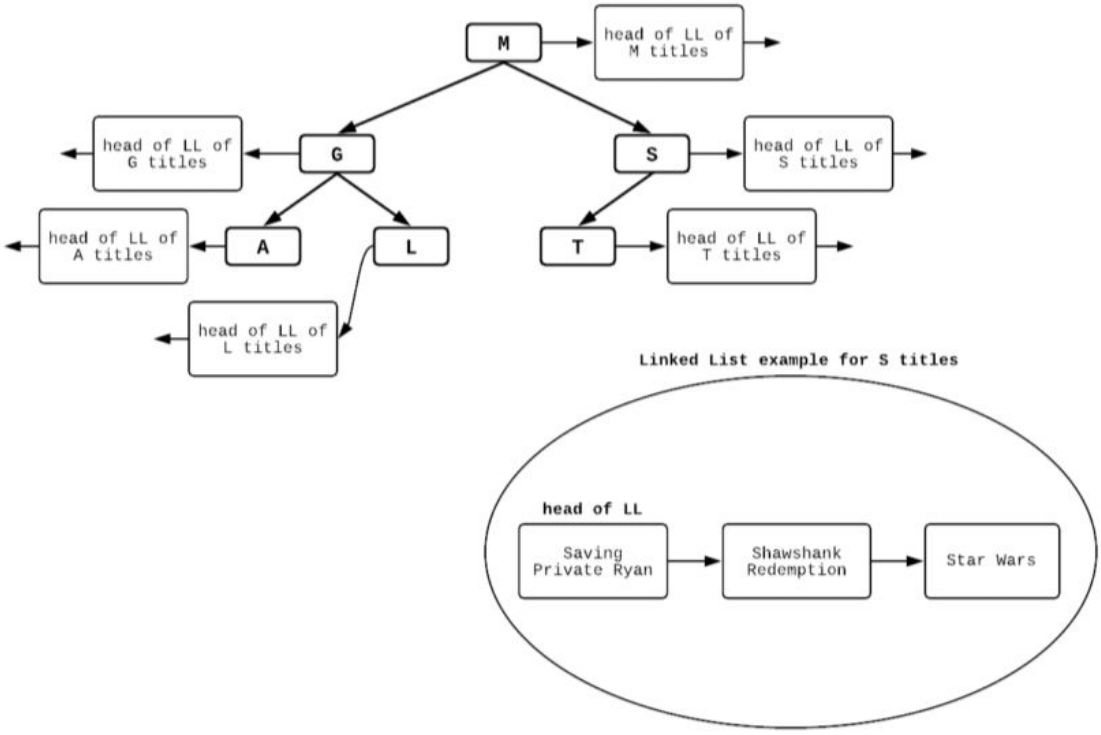
When the user selects this option, your program should delete the nodes in the tree and exit the program.

When the user selects quit, the destructor for the MovieTree class should be called and in the destructor, all of the nodes in the tree and singly linked lists should be deleted. You need to use a *postorder* tree traversal for the delete or you will get segmentation fault errors.

Use the cout statements in Appendix A to set the order of the menu options.

What does a BST of Linked Lists look like

Here's a diagram to help you visualize the data structure you need to build for this assignment. The BST nodes are marked with individual letters as the key for the node. Each BST node then contains a pointer to a singly linked list for the titles that start with that letter. Titles in the linked list need to be in sorted order, ascending.



Implementation

Your BST should be implemented in a class. For this assignment, create a header file *MovieTree.hpp* that creates the class prototypes necessary for the assignment. Implement the functionality of the class in a corresponding *MovieTree.cpp* file and implement the functionality of the program in *Assignment4.cpp*. Your implementation files should be set up so they can be compiled with:

g++ -std=c++11 MovieTree.cpp Assignment3.cpp -o Assignment4

and then run your program on the command-line using

./Assignment4 Movies.txt

Your class should include methods for inserting movies, searching for movies, renting movies and deleting movie nodes, as well as methods for printing and counting all the movies in the tree. Additionally the class will need constructor and destructor methods.

Submission

Your solution to Assignment 4 should be submitted to moodle with two parts: source code and a readme. Include the source code files you wrote to implement the solution to this assignment. Make sure your code is commented enough to describe what it is doing. Include a comment block at the top of the all files with your name, assignment number, and course instructor and section. Your submission should also include a brief and informative readme.

Appendix A – cout statements

Display menu

```
cout << "======Main Menu======" << endl;
cout << "1. Find a movie" << endl;
cout << "2. Rent a movie" << endl;
cout << "3. Print the inventory" << endl;
cout << "4. Delete a movie" << endl;
cout << "5. Count the movies" << endl;
cout << "6. Quit" << endl;
```

Find a movie

```
cout << "Enter title:" << endl;
```

Display found movie information

```
cout << "Movie Info:" << endl;
cout << "===========" << endl;
cout << "Ranking:" << foundMovie->ranking << endl;
cout << "Title:" << foundMovie->title << endl;
cout << "Year:" << foundMovie->year << endl;
cout << "Quantity:" << foundMovie->quantity << endl;
```

If movie not found

```
cout << "Movie not found." << endl;
```

Rent a movie

```
//If movie is in stock
cout << "Movie has been rented." << endl;
cout << "Movie Info:" << endl;
cout << "===========" << endl;
cout << "Ranking:" << foundMovie->ranking << endl;
cout << "Title:" << foundMovie->title << endl;
cout << "Year:" << foundMovie->year << endl;
cout << "Quantity:" << foundMovie->quantity << endl;

//If movie not found in tree
cout << "Movie not found." << endl;
```

Print the inventory

```
//For all movies in tree
cout<<"Movie: "<<node->title<<" "<<node->quantity<<endl;
```

Count movies in the tree

```
cout<<"Tree contains: "<<mt.countMovieNodes()<<" movies." << endl;
```

Delete movie

```
cout << "Enter title:" << endl;
//If movie not found in tree
cout << "Movie not found." << endl;
```

Delete all nodes in the tree

```
//For all movies in tree
cout<<"Deleting: "<<node->title<<endl;
```

Quit

```
cout << "Goodbye!" << endl;
```