CSCI 2270 Data Structures and Algorithms Summer 2017
Instructor: Holler/MonteroQuesada
Assignment 5

Due Friday, July 7th, by 11am Assignment 5 Part A (Pseudocode)
Due Wednesday, July 12th, by 8am Assignment 5 Part B (Implementation)

Binary Search Trees

Objectives.

• Create, add, and delete, and manipulate a BST

Introduction.

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 movie title. 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 previous assignments 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 movie title. All other information about the movie should also be included in the node for that movie in the tree.

Note: the data should be added to the tree in the order it is read in. The name of the file that contains the movie data is Assignment5Movies.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 display all information for that movie. If the movie is not found in the tree, 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 the tree, 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 tree.

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. See the lecture notes and recitation exercises on in-order tree traversal 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 that movie, delete it if it’s found, re-assign to 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 in any order 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 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.

Implementation details
There are several components to this assignment that can be treated independently. Our advice is to tackle these components one by one, starting with updating the menu from the last assignment to meet the requirements for this assignment.

Although you need to submit a single code for the implementation and driver, while developing your code you should practice working with three separate files - MovieTree.hpp, MovieTree.cpp, and Assignment5.cpp. You need to write the MovieTree.hpp, MovieTree.cpp, and Assignment5.cpp files. You can compile your code on the command-line using g++

```
g++ -std=c++11 MovieTree.cpp Assignment5.cpp -o Assignment5
```
and then run your program on the command-line using

```
./Assignment5 <name_of_the_movie_file>
```

Also, remember to start early.

Pseudocode (20%)
For this assignment, the pseudocode portion will correlate to the functionality of node deletion in a BST tree.

Implementation (80%)
Submit your Assignment 5 code by opening the link to the CodeRunner question and pasting your code on the text box. Make sure your code is commented enough to describe what it is doing. Include a comment block at the top with your name, assignment number, and course instructor.

If you do not get your assignment to run on CodeRunner, you will have the option of scheduling an interview grade with your TA to get a grade for the assignment. Even if you do get the assignment to run on CodeRunner, you can schedule the interview if you just want to talk about the assignment and get feedback on your implementation. Consider the TA office hours if you think that the discussion would be longer than 10min.

What to do if you have questions
There are several ways to get help on assignments in 2270, and depending on your question, some sources are better than others. The Piazza discussion forum is a good place to post technical questions, such as how to add a node to a linked list. When you answer other

students' questions on the forum, please do not post entire assignment solutions. The CAs and TAs are also a good source of technical information, especially questions about C++. If, after reading the assignment write-up, you need clarification on what you're being asked to do in the assignment, the TAs and the Instructor are better sources of information than the discussion forum or the CAs.

Appendix A – cout statements that CodeRunner expects

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;
```