CSCI 2270 – Data Structures and Algorithms
Instructor: Hoenigman
Assignment 7
Due Wednesday, March 9 by 3pm

Binary Search Trees – Part Two

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 from assignment 6. Use the same Assignment6Movies.txt file that you used for assignment 6 and pass the name of the file as a command line argument.

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. 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 in the tree. *Note: the data should be added to the tree in the order it is read in.*

1. **Find a movie.**
   When the user selects this option from the menu, they should be prompted for the name of the movie. You 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."
2. **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.

3. **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 on in-order tree traversal for more information.
4. **Quit the program.**
   When the user selects this option, your program should delete the nodes in the tree and exit the program.

Additional Functionality for Assignment 7

1. **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 the quantity goes to 0 for any movie.

2. **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.

3. **Delete the tree in the destructor using a postorder traversal.**
   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.

Incorporate this functionality into the menu you display. Use the cout statements in Appendix A to set the order of the menu options.

Implementation details

Your BST should be implemented in a class. You are provided with a MovieTree.h file on Moodle that includes the class prototype for this assignment. You need to implement the class functionality in a corresponding MovieTree.cpp file and Assignment7.cpp file. To submit your work, zip all files together and submit them to COG. If you do not get your assignment working on COG, you will have the option of a grading interview.

Appendix A – cout statements that COG 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;
```