CSCI-2270
Assignment 6
Instructor: Boese

Binary Search Tree

Online Movie Service – Part 2

Objectives

- Delete a node in a BST
- Traverse a BST
- Delete a BST

Background

Online Movie Service (From Assignment 5)
You must have everything from Assignment 5 working to complete this assignment. Make a copy of your Assignment 5 code to use as a starting point for this assignment.

Program Specifications

- **Add the following functionality to your Assignment 5 Binary Search Tree implementation:**

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 node, remove it from the tree, re-assign pointers to bypass the removed node, and free the memory used by the node. **If the node to be deleted has both left and right children, replace the deleted node with the node with the minimum value in its right subtree.** 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.
   Be sure to test your program for movie nodes with 0 children, 1 child, or 2 children!

2. Count movies in the tree.
   When the user selects this option, your program should traverse the tree in any order, count the total number of 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. 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. You may implement a helper function.

- **Menu**: After the tree has been built, display a menu with the following options:

(includes functionality from Assignment 5).

1. 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." The `findMovie` method included in the MovieTree.hpp header file is a void type. All movie information should be displayed in the `findMovie` method. This should already be implemented from Assignment 5.

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." Just like findMovie, rentMovie is also a void type. Information about the movie rented should be displayed in the rentMovie method. If the quantity of a movie reaches 0, delete the movie after displaying the updated movie information. This should be **mostly** implemented from Assignment 5.

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 and your textbook for more information. This should already be implemented from Assignment 5.

4. Delete 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 delete the movie node from the tree. If the movie is not found, your program should display, "Movie not found."

5. Count movies in the tree. When the user selects this option, your program should traverse the tree, counting the total number of movie nodes in the tree, and print the count.

6. Quit the program. When the user selects this option, your program should exit after freeing all dynamically allocated memory.

- **Expected Output**

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 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;
```

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

Count movies
```
cout << "Count = " << count << endl;
```

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

MovieTree.hpp Header File:

```
#ifndef MOVIETREE_HPP
#define MOVIETREE_HPP

#include <string>
struct MovieNode
{
      int ranking;
      std::string title;
      int year;
      int quantity;
      MovieNode *parent;
      MovieNode *leftChild;
      MovieNode *rightChild;

      MovieNode(){};

      MovieNode(int in_ranking, std::string in_title,
      int in_year, int in_quantity)
      {
             ranking = in_ranking;
             title = in_title;
             year = in_year;
             quantity = in_quantity;
             parent = leftChild = rightChild = nullptr;
       }
};

class MovieTree
{
public:
       MovieTree();
       ~MovieTree();
       void printMovieInventory();
       void addMovieNode(int ranking, std::string title,
                         int releaseYear, int quantity);
       void findMovie(std::string title);
       void rentMovie(std::string title);
       void deleteMovie(std::string title);
       void countMovies();


private:
       MovieNode *search(MovieNode *node, std::string title);
       MovieNode *root;
};
#endif // MOVIETREE_HPP
```

Submitting Your Code:

Log into Moodle and go to the Homework 6 link. It is set up in quiz format. Follow the instructions on each question to submit all or parts of your code. You can check your solution to each question by clicking on the "Check" button multiple times without penalty. If you wrote helper functions, you may use them in the same

answer boxes as your method implementations. Simply copy the helper function definition before the definition of the method.

Note that you can only submit your homework once.

Note: there is no late period on assignments! If you miss the deadline or do not do well, you can sign up for an optional grading interview to get up to half the points missed back. There is also an optional extra credit assignment at the end of the semester you can use to replace one of your homework scores.