CSCI-2270
Assignment 5
Instructor: Boese

Binary Search Tree

Online Movie Service – Part 1

Objectives

- Build a binary search tree (BST)
- Search and traverse a BST

Background

Online Movie Service

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, and quitting the program.

Program Specifications

- **Movie Information** is in a file and the name of the file is passed in as a command-line argument. An example file is on the website HW5-Movies.txt.

- **Insert all the movies in the tree.** Open the movies file and read all movie data line-by-line, creating a MovieNode for each movie. Insert each node into the Binary Search Tree ordered alphabetically by movie title. *Note: the data should be added to the tree in the order it is read in.*

- You must implement the methods and make use of the Binary Search Tree defined in the MovieTree.hpp file (below).

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

- 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." If the movie is found in the tree, but the Quantity is zero, display "Movie out of stock.". Just like findMovie, rentMovie is also a void type. Information about the movie rented should be displayed in the rentMovie method.

- 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. *If the inventory is empty, do not print anything.*

- 4. Quit the program. When the user selects this option, your program should exit.

- **Hints**
  - The header file requires a destructor `~MovieTree()` but we will not test this function in this assignment. A future assignment may test your ability to write a correct destructor for a Tree.
  - The private method:

    ```
    MovieNode *MovieTree::search(MovieNode *node, std::string title)
    ```

    should search the tree pointed to by `node` for a node with the given title. We will not test this method directly, but you may call this private `search` method from the other methods in the MovieTree class.

- **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. 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
```
cout << "Enter title:" << endl;

//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 is out of stock
cout << "Movie out of stock." << 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;
```

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);

    private:
        MovieNode *search(MovieNode *node, std::string title);
        MovieNode *root;

};

#endif // MOVIETREE_HPP
```

Submitting Your Code:

Log into Moodle and go to the Homework 5 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.