CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 5, Oct 2018

Binary Search Trees

OBJECTIVES

1. **Build a binary search tree (BST)**
2. **Search and traverse a BST**

Background

Imagine an online movie service needs help keeping track of their stock. They have a large number of movies, and want you to help them by developing a program that stores the movies in a binary search tree for fast access. The movies will be accessed by their titles, but they will also store the following information on each one:

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

Your program will have a menu similar to assignment 2, which will let the user select the options they want to perform. In this assignment, the menu will let the user find information on a movie, rent a movie, and print the entire movie inventory.

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 5, Oct 2018

Assignment

- Your code should implement a binary search tree of movies. A header file that lays out this tree can be found in *HW5-MovieTree.hpp* on Moodle. To pass autograding, **do not** change the class definition in the header file, but **you can** add additional helper functions as you see fit. You will need to implement the following functions:

    - ***`MovieTree()`*** **and** ***`~MovieTree()`***
      The constructor should initialize any member variables of the class, while the destructor should free any dynamic memory that was allocated to create the tree.

      Hint: You may need to create a helper function for the destructor to use.

    - ***`void printMovieInventory()`***
      This function should print out every node in the tree, in alphabetical order. It should print in the following format:

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

      Hint: You may need to create a helper function.

    - ***`void addMovieNode(int ranking, string title,`***
      `int year, int quantity)`
      This function should add a node to the tree in the correct place based on its title. Every node's left children should come alphabetically **before** that node, and its right children should come alphabetically **after** that node.

- ***void findMovie(string title)***
  This function should find the specified movie in the tree, then print out its information in the following format:

```
// 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 the movie isn't found, print the following instead:

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

  Hint: Using the search function defined below can make this much easier to write!

- ***void rentMovie(string title)***
  This function should find the specified movie in the tree, then rent it if there are any remaining in stock by decreasing the quantity by 1. If it succeeds, it should print out that the movie has been rented, *then* print the same information that gets printed when a movie is found, in the following format:

```
cout << "Movie has been rented." << endl;
// Also print the same information as when the movie is found
```

  If the movie isn't found, print the following instead:

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

  If the movie is found, but does not have enough copies to rent, print the following:

```
cout << "Movie out of stock." << endl;
```

  Hint: Using the search function defined below can make this much easier to write!

- ***MovieNode *search(string title)***
  This is a helper function meant to be used in *rentMovie* and *findMovie*. It should return a pointer to the movie with the given title, or NULL if no such movie exists.

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 5, Oct 2018

- Your main function should first read information about each movie from a file and store that information in a MovieTree. **The name of the file with this information should be passed in as a command-line argument.** An example file is *HW5-Movies.txt* on Moodle. It is in the format:

```
1,<Movie 1 title>,<Movie 1 year>,<Movie 1 quantity>
2,<Movie 2 title>,<Movie 2 year>,<Movie 2 quantity>
Etc…
```

  Note: For autograding's sake, add the nodes to the tree in the order they are read in.

- **After** reading in the information on each movie from the file, display a menu to the user. The code to print this menu is below:

```
// 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;
```

- The options should have the following behavior:
  - **Find a movie**
    Call your tree's *findMovie* function on a movie specified by the user. Prompt the user for a movie title using the following code:

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

  - **Rent a movie**
    Call your tree's *rentMovie* function on a movie specified by the user. Prompt the user for a movie title using the following code:

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

  - **Print the inventory**
    Call your tree's *printMovieInventory* function.
  - **Quit**
    Exit after printing a friendly message to the user:

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

Submission

Submit your code on Moodle by following the directions on Assignment 4 Submit. You must submit an answer to be eligible for interview grading!