CSCI 2270 – Data Structures

Assignment 6

Binary Search Tree

OBJECTIVES

Building a binary search tree (BST)
Finding a node in BST
Traversing trees.

Background

Assume that you are given the task of predicting user ratings given a dataset of TV shows. To accomplish this goal, build a Binary Search Tree (BST) that can be used to search for shows and extract their features in an efficient manner. The shows should be accessible by their titles, but they will also store the following features:

- Title
- Year released
- Show rating (G, PG, PG-13, R, etc.)
- User rating

Your Binary search tree will utilize the following struct with default and overloaded constructors:

```
struct ShowItem {
      string title;
      int year;
      string showRating;
      float userRating;
      ShowItem* left = NULL;
      ShowItem* right = NULL;

      ShowItem(string t, int y, string sr, float ur) {
            title = t;
            year = y;
            showRating = sr;
            userRating = ur;
      }
};
```

ShowCatalog Class

Your code should implement a binary search tree of ShowItem data type. A header file that lays out this tree can be found in *ShowCatalog.hpp*. As usual, **DO NOT** modify the header file. *You may implement helper functions in your .cpp file to facilitate recursion if you want as long as you don't add those functions to the ShowCatalog class*.

ShowCatalog()

➔ Constructor: Initialize any member variables of the class to default

~ShowCatalog()

➔ Destructor: Free all memory that was allocated

void printShowCatalog()

➔ Print every node in the tree using Preorder traversal of titles using the following format:

```
// for every Show node (s) in the tree
cout << "Show: " << s->title << " " << s->userRating <<
endl;
```

If there is no show entry in the tree, print the following message instead:

```
cout << "Tree is Empty. Cannot print" << endl;
```

void addShowItem(string title, int year, string showRating, float userRating)

➔ Add a node to the tree in the correct place based on its **title**. Every node's left children should come before it alphabetically, and every node's right children should come after it alphabetically. *Hint: you can compare strings with <, >, ==, string::compare() function.*

➔ *For example*, if the root node of the tree is the show "Boys", then the show

"Stranger Things" should be in the root's right subtree and "Arcane: League of Legends" should be in its left subtree.

➔ You can assume that no two shows have the same title

void getShow(string title)

➔ Find the show with the given **title**, then print out its information:

```
cout << "Show Info:" << endl;
cout << "==================" << endl;
cout << "Title :" << node->title << endl;
cout << "Year :" << node->year << endl;
cout << "Show Rating :" << node->showRating << endl;
cout << "User Rating :" << node->userRating << endl;
```

If the show isn't found, print the following message instead:

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

void searchShows(char titleChar)

➔ Print all the shows whose title starts with the **titleChar** character using the following format:

```
cout << "Shows that starts with " << titleChar << ":"
<< endl;
 // each show that satisfies the constraints should be printed with
cout << s->title << "(" << s->year << ") " << s->userRating << endl;
```

If there is no show entry in the tree, print the following message instead:

```
cout << "Tree is Empty. Cannot search Shows" << endl;
```

For example, consider your BST has shows

1. Adventure Time
2. Big Little Lies
3. Dark

searchShow('A') should print all the shows starting with character 'A'. So, calling searchShow('A') for the above BST will print "Adventure Time".

void displayNumShowRating(int &count, string showRating)

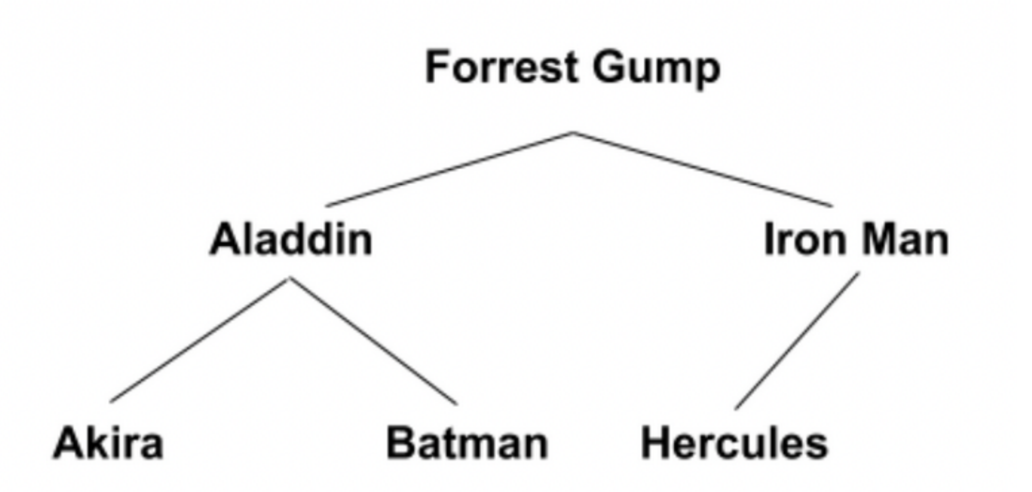
➔ Change the reference variable count to have count of all shows with **showRating** given in the parameters.

void printLeafNodes()

➔ Print all the leaf nodes of the BST using the following format:

```
    cout << root->title << endl;
```

E.g: In the following BST, Akira, Batman, Hercules are leaf nodes.



Driver

For this assignment, the driver code has been provided to you in the app_1/*main_1.cpp* file. The main function will read information about each show from a CSV file (use shows.csv) and store that information in a ShowCatalog using your **addShowItem** function implementation.

The name of the CSV file with this information should be passed in as a command-line argument. An example file, *shows.csv*, has been provided to you. It contains the following format:

```
<Show 1 title>,<Show 1 year>,<Show 1 show rating>,<Show 1
user rating>
<Show 2 title>,<Show 2 year>,<Show 2 show rating>,<Show 2
user rating>
Etc...
```

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

After reading the information on each show from the file and building the tree, the user is displayed the following menu:

```
======Main Menu======
  1. Get a show
  2. Search shows
  3. Print show catalog
  4. Display number of show ratings
  5. Print Leaves
  6. Quit
  >
```

- **Get a show:** Call your tree's **getShow** function on a show specified by the user. The user is prompted for a show title.

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

- **Search shows:** Calls your tree's **searchShow** function on a title character specified by the user. Prompts the user for the title character.

```
cout << "Enter the title character:" << endl;
// get user input
```

- **Print the catalog:** Call your tree's **printShowCatalog** function
- **Display number of show ratings:** calls **displayNumShowRating**(int &count, string showRating) function. Prompts the use for showRating to count the number of.

```
cout << "Enter the show rating to count:" << endl;
// get user input - PG-13, PG, R
cout << "Number of " << rating << ": " << count << endl;
// Beware that count is printed from reference variable passed.
```

- **Print Leaves:** Call your tree's **printLeafNodes** function
- **Quit:** Program exits after printing a friendly message to the user.

CSCI 2270 – Data Structures

For running run_app_1(Which runs main_1.cpp):

```
./run_app_1 ../<showsFileName>
```

Order of function implementation

- ➔ Destructor and Constructor
- ➔ addShowItem and printShowCatalog
- ➔ getShow
- ➔ searchShows
- ➔ displayNumShowRating
- ➔ printLeafNodes