EECS 560 Lab 4 – Implementation of Binary Tree

Prof.: Dr.Shontz, GTAs: Chiranjeevi Pippalla, Prashanthi Mallojula

100 Points

Due date:
11:59pm, Monday, 02/24/2020 for Tuesday lab.
11:59pm, Wednesday, 02/26/2020 for Thursday lab.

Purpose:
The purpose of this lab is to implement a binary tree in C++.

General Requirements:
In this lab, you are required to implement a binary tree **using a pointer implementation** (do not use an array). Each node of the tree will have a key and left and right pointers, where the left pointer will point to its left child, and the right pointer will point to its right child. You are to read in a collection of movie records from a data file called data.txt in **level-order**. Each record consists of two fields – movie title and movie rating. The movie name should be of type string, and the movie rating should be of type float. The rating field should neither be less than 0 nor greater than 5. **Duplicate records are not allowed in the binary tree**. You may hard code the file name if you wish.

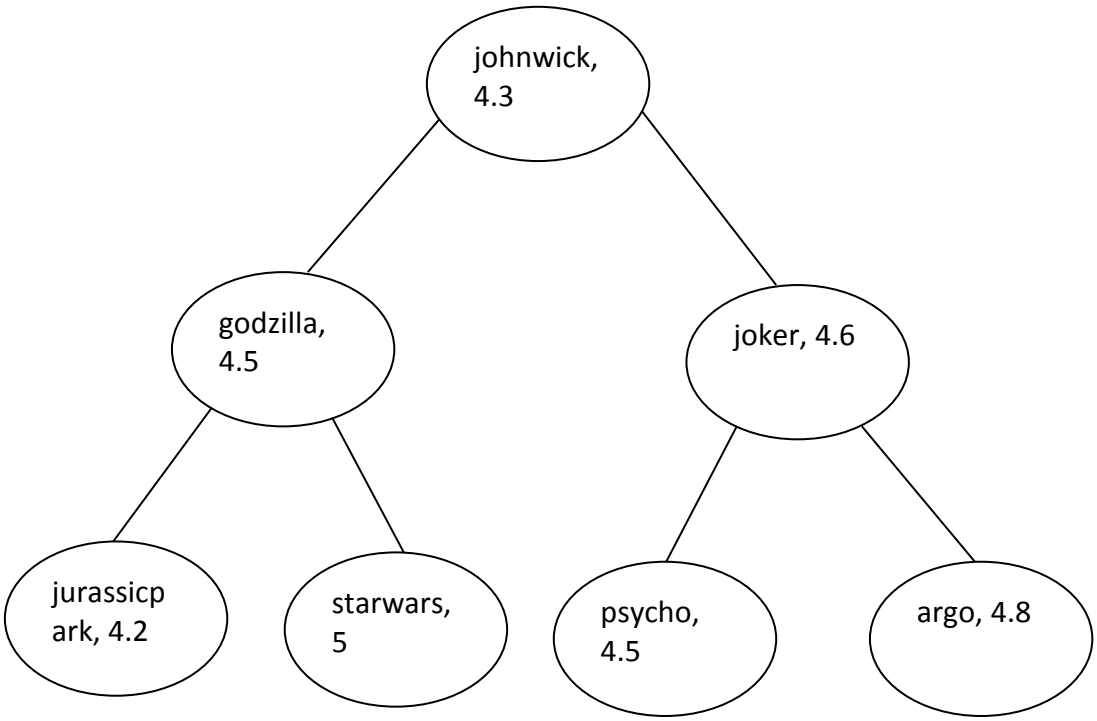

Fig. 1

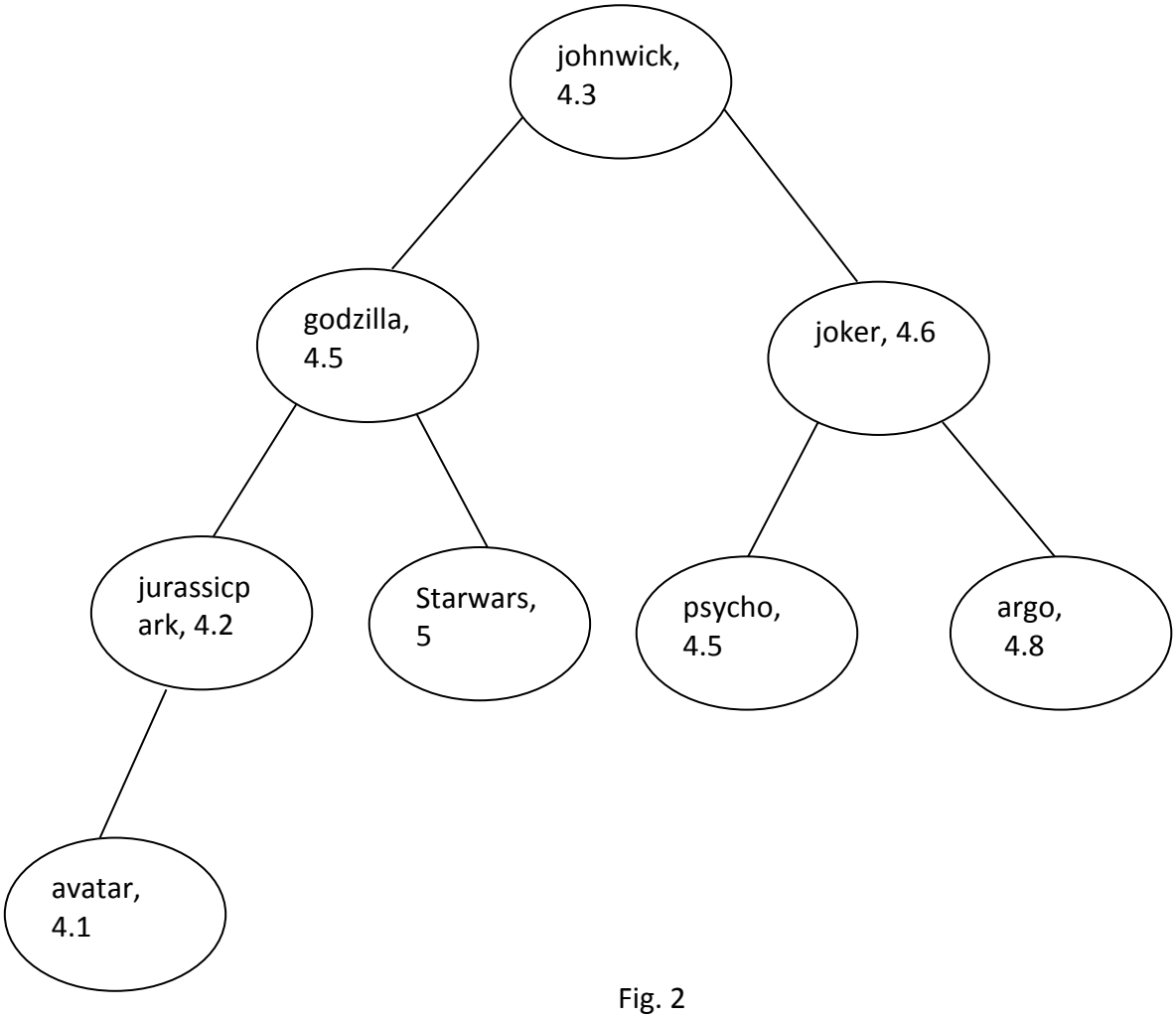

Fig. 2

The binary tree should be implemented with an appropriate constructor and destructor. The rest of the methods should be implemented as follows:

- **IsFull()** – This function should return true if the tree is a full tree. Only Print 'True' or 'False' in the output. To check the correctness of the function, test this function twice, i.e., once after reading in the values and constructing the tree from data.txt and again after adding a movie record.
- **AddMovie(title, rating)** – This function adds a movie record to the binary tree in a level-order fashion. Every new record should be added in the next position and the function should print the statement "Record has been added successfully".
- **RemoveMovie(**) – This function deletes the record that occurs at the last position according to a level order traversal in the binary tree and should print the statement "The last movie with title "title" and rating "rating" has been deleted "(movie name and rating) that it has deleted.
- **Leaf(title)** – This function will check whether or not the node containing the given movie title has a child. If it does not have a child, the node/record is a leaf node. In that case,

the function should return true. The output should be: The node containing the movie title “title” is/isn’t a leaf node.

- **PrintLeaves()** – This function prints the leaf nodes one after the other. The order of traversal/printing should be in level order.
- **PrintTreeHeight()** – Prints the height of the tree.
- **Preorder()** – Traverses the tree in pre-order and prints the movie ratings all of the records.
- **Postorder()** – Traverses the tree in post-order and prints the movie ratings all of the records.
- **Inorder()** – Traverses the tree in in-order and prints the movie ratings all of the records.
- **Levelorder()** – Traverses the tree in level order and prints the movie titles all of the records.

Examples for reference:
These are some examples related to the operations mentioned above:

- **IsFull():** The tree shown in **Fig. 1** is a full binary tree, whereas the tree shown in **Fig. 2** is not a full binary tree.
- **AddMovie(title, rating):** The binary tree should look like **Fig. 1**, and adding a record (say Harry Potter, 4.3) will change the structure of the binary tree to **Fig. 2**.
- **RemoveMovie ():** The function if called on the tree shown in **Fig. 2** should alter the state of the tree to **Fig. 1**.

Expected Results:
data.txt:

johnwick, 4.3
godzilla, 4.5
joker, 4.6
jurassicpark, 4.2
starwars, 5
psycho, 4.4,
argo, 4.8

After building the tree by reading the data.txt in **level order**, your program should ask the user to choose one of the options below:

*Please note that the outputs shown below are only for your reference. The actual outputs may differ.

Please choose one of the following commands:
1- IsFull

2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>1
>Output: True

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>8
> Output: Printing in Post-Order – 4.2, 5, 4.5, 4.5, 4.8, 4.6, 4.3

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>2
> Please enter the movie title which you want to enter into the tree.
>avatar
> Please enter the rating of your movie

> 4.1
> Output: Record has been added successfully!

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>10
> Output: Printing in Level-Order – johnwick, godzilla, joker, jurassicpark, starwars, psycho, argo, avatar
---
Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>4
> Please enter the name of your movie which you want to test as a leaf node.
> argo
>Output: The record with movie title argo is a leaf node.

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder

8- Postorder
9- Inorder
10- Levelorder
11- Exit
>3
> The last movie with title avatar and rating 4.1 has been deleted.

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>5
>Output: The leaf nodes are: jurassicpark, starwars, psycho, argo.

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>6
> Output: The height of the tree is 2.

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight

7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>7
>Output: Printing the tree in Pre-Order – 4.3, 4.5, 4.2, 5, 4.6, 4.5, 4.8

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- PrintTreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>9
> Output: Printing the tree in In-Order – 4.2, 4.5, 5, 4.3, 4.5, 4.6, 4.8

Please choose one of the following commands:
1- IsFull
2- AddMovie
3- RemoveMovie
4- Leaf
5- PrintLeaves
6- TreeHeight
7- Preorder
8- Postorder
9- Inorder
10- Levelorder
11- Exit
>11
> Output: Goodbye!

Execution Instructions:

GDB/DDD can be used to debug your code. Make sure to use Valgrind to execute your code to check for any memory leaks/segmentation faults.

Grading rubric:

- Full grade: Program should execute without any issues with all the options executed and with no memory leaks.
- Points will be taken for any execution errors, such as memory leaks, segmentation/program abort issues and missing handling of invalid cases.
- Programs that are compiled but do not execute will earn in the range of 0 to 50% of the possible points. Your grade will be determined based on the program design and the options implemented in the code.

Submission instructions:

- All files, i.e., the source files and Makefile, should be zipped in a folder.
- Include a ReadMe.txt if your code requires any special instructions to run.
- The naming convention of the folder should be LastName_Lab4.zip (or .tar or .rar or .gz).
- Email it to your respective lab instructor: chiranjeevi.pippalla@ku.edu (Chiru) or prashanthi.mallojula@ku.edu (Prashanthi) with subject line EECS 560 Lab4.
- Your program should compile and run on the **Linux machines** in **Eaton 1005D using g++.**