Peter M Piper

This program implements an AVL tree as an example of balanced binary search tree. It displays the tree with its balance factors.

Data Structures 2

AVL Tree

**Reason for Test Case \_\_\_\_\_\_\_\_ Input Values\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Expected Output**

Display tree – without nodes tree is empty, “The tree is empty!”

select display tree

Display Tree – int nodes create tree: 1, 2, 3 2(0)

/ \

1(0) 3(0)

InOrder: 1 2 3

PreOrder: 1 2 3

PostOrder: 3 2 1

Display Tree – string nodes create tree: The, Hunger, Games Hunger(0)

/ \

Games(0) The(0)

InOrder: Games Hunger The

PreOrder: The Hunger Games

PostOrder: Games Hunger The

Display Tree – with string nodes create tree: The, hunger, Games The(0)

/ \

Games(0) hunger(0)

InOrder: Games The hunger

PreOrder: The Games hunger

PostOrder: Games hunger The

**Reason for Test Case \_\_\_\_\_\_\_\_ Input Values\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Expected Output**

Display Tree – balance factor/rotations create tree: A, B A(1)

\

B(0)

Insert C B(0)

/ \

A(0) C(0)

Insert D B(1)

/ \

A(0) C(1)

\

D(0)

Remove A C(0)

/ \

B(0) D(0)

Insert node – single digit int select insert integer, enter 3 “3 is inputed!”

Insert node – multiple digit int select insert integer, enter 123 “123 is inputed!”

Insert node – invalid int select insert integer, enter A “A is not a valid number!”

Insert node – duplicate int select insert integer, 2 then 2 “2 already exists!”

**Reason for Test Case \_\_\_\_\_\_\_\_ Input Values\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Expected Output**

Insert node – int empty value select insert integer, enter without input “ is not a valid number!”

Insert node – single char select insert string, enter A “A is inputed!”

Insert Node - string select insert string, enter Hunger “Hunger is inputed”

Insert Node – duplicate string select insert string, Hunger then Hunger “Hunger already exists!”

Insert Node – different case select insert string, Hunger then hunger “hunger is inputed”

Insert Node – string empty value select insert string, enter without input “ is not a valid string!”

Remove node – int exists create tree with 1, 2, 3. “1 is removed!”

select remove node, input 1

Remove node – int does not exist create tree with 1, 2, 3. “Number was not found!”

select remove node, input 6

Remove node – string exists create tree with a, b, c. “a is removed!”

select remove node, input a

Remove node – string does not exist create tree with a, b, c “String was not found!”

select remove node, input f

**Reason for Test Case \_\_\_\_\_\_\_\_ Input Values\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Expected Output**

Search for node – int exist create tree with 1, 2, 3 “1 was found!”

select search for node, input 1

Search for node – int does not exist create tree with 1, 2, 3 “6 was not found!”

select search for node, input 6

Search for node – int empty value select search for node, “ is not a valid number!”

enter without input

Search for node – string exists create tree with a, b, c “a was found!”

select search for node, input a

Search for node – string does not exist create tree with a, b, c “f was not found”

select search for node, input f

Search for node – string empty value select search, enter without input “ is not a valid string!”

Get root – empty tree select get root “The root does not exist!”

Get root –tree created create tree with a, b, c “The root item is: a”

select get root

Chop tree down select chop tree “Chopping…Timber!”

“Tree is empty!”

**Reason for Test Case \_\_\_\_\_\_\_\_ Input Values\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Expected Output**

New tree select new Tree tree is chopped

Full tree test - full select full tree test “The tree is full!”

Full tree test – not full select full tree test “The tree is not full!”

Quit program select quit program Exits program

