Metropolitan State University

ICS 240 - 01: Introduction to Data Structures

Worksheet Wb: Trees

Total Points: 10

Out: We Apr 2, 2025

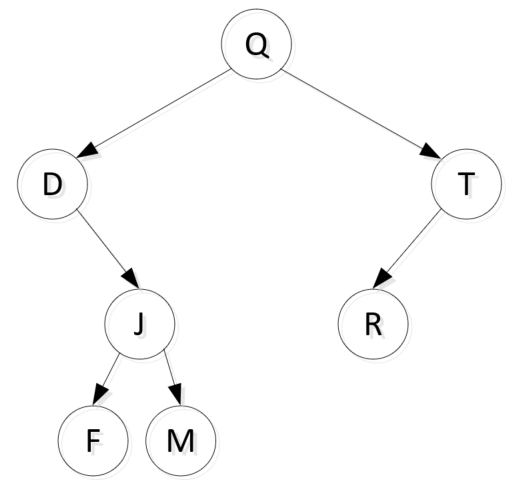
Due: Sat Apr 5, 2025

## What to submit?

Upload only ONE file to the Worksheet-a D2L folder. Some questions may be answered directly on the file. Other questions can be answered on a piece of paper, and you take a picture of your paper and insert the picture in the word file.

# Exercise 1: Binary Search Tree Traversals

[3 pts] List the preorder, inorder, and postorder traversals for this tree.



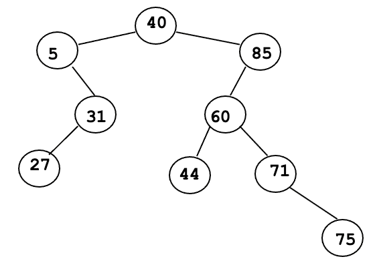
Preorder: Q D J F M T R

Inorder: D F J M Q R T

Postorder: F M J D R T Q

# Exercise 2: Binary Search Tree Construction Puzzle

[2 pts] Consider the following Binary Search Tree.



Assume you want to store the numbers in the given tree is a **complete binary search tree**.

What would a complete binary tree with these numbers look like? Draw the tree below:

40

/ \

5 85

/ \

31 60

/ / \

27 44 71

/ \

68 75

In what order should you insert the numbers so that the resulting tree is complete?

List the numbers in the order in which you would add them, preferably comma-separated.

40, 5, 31, 27, 85, 60, 44, 71, 68, 75

# Exercise 3: Using a simple class IntBTNode

[2 pts] Consider the very basic IntBTNode.java class shown below. A Java source code file with this class is on the D2L site under Worksheets 🡪 Worksheet a. Add a main method to this class that will create the small binary tree below (note that it is not a binary search tree, and that’s OK), and print the preorder, inorder, and postorder traversals of this tree. Include a screen shot of your main method and the corresponding output. Don’t upload a separate code file. Just a screenshot of the main method, and the output.

14

/ \

2 11

/ \

3 30

public class IntBTNode {

private int data;

private IntBTNode left;

private IntBTNode right;

public IntBTNode( int data, IntBTNode left, IntBTNode right ) {

this.data = data;

this.left = left;

this.right = right;

}

public int getData() { return data; }

public IntBTNode getLeft() { return left; }

public IntBTNode getRight() { return right; }

public void setData( int data ) { this.data = data; }

public void setLeft( IntBTNode left ) { this.left = left; }

public void setRight( IntBTNode right ) { this.right = right; }

public boolean isLeaf() {

return ( left == null ) && ( right == null );

}

public void preorder() {

System.out.print( getData() + " " );

if ( getLeft() != null ) {

getLeft().preorder();

}

if ( getRight() != null ) {

getRight().preorder();

}

}

public void inorder() {

if ( getLeft() != null ) {

getLeft().inorder();

}

System.out.print( getData() + " " );

if ( getRight() != null ) {

getRight().inorder();

}

}

public void postorder() {

if ( getLeft() != null ) {

getLeft().postorder();

}

if ( getRight() != null ) {

getRight().postorder();

}

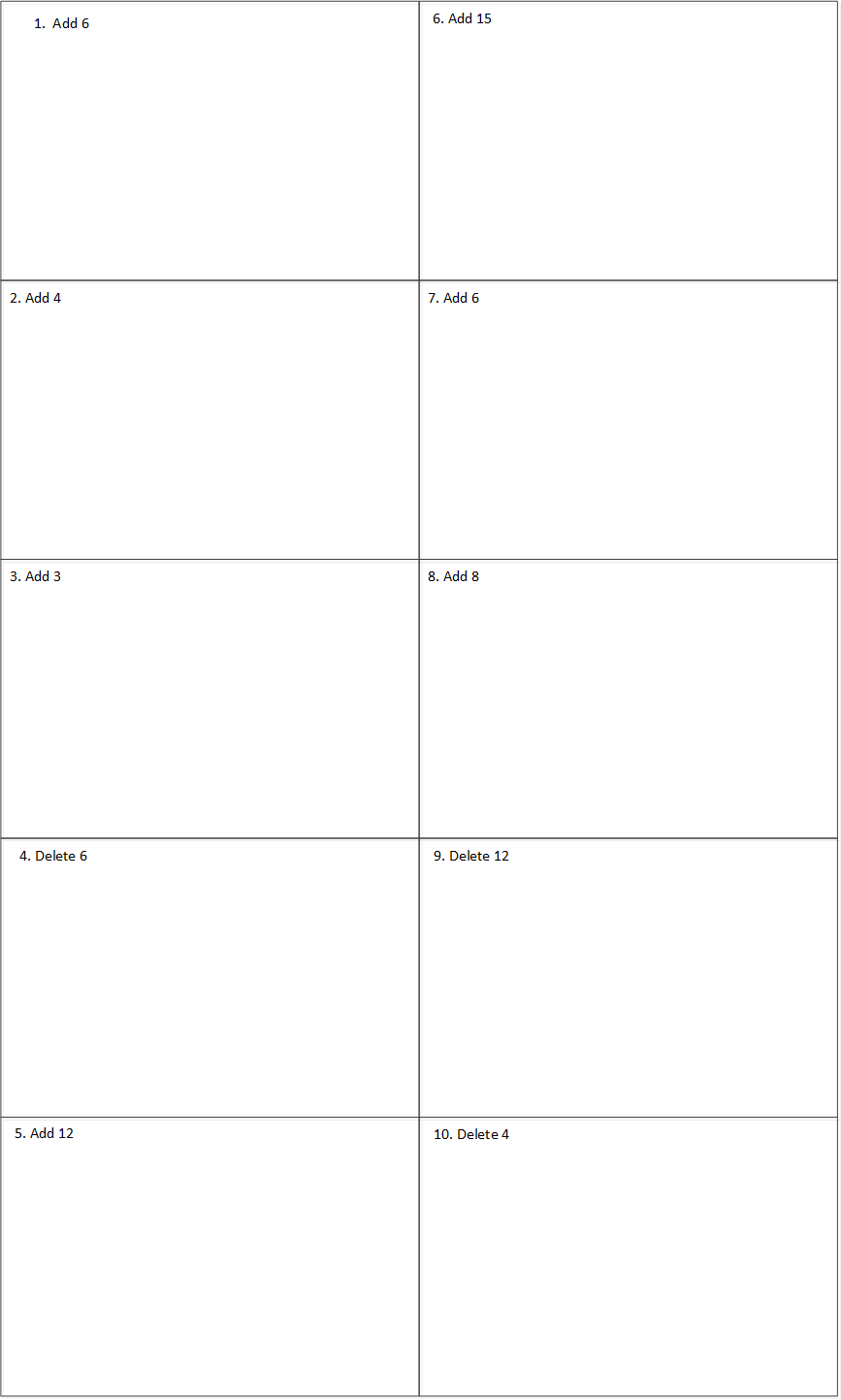
System.out.print( getData() + " " );

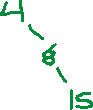
}

}

# Exercise 4: Trace BST adds and deletes

Draw the Binary Search Tree that results from each of the following adds and deletes in order. Note that the order of operations is by columns. See the appendix for a simple example of what I’m looking for. Do deletes as in the slides, replacing the element with the rightmost element of the left subtree (if one exists).





## Appendix: Drawing BST adds and deletes, an example

Here is an example of the format I want you to use when answering Exercise 4.

