Name:\_Jeremiah Webb\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CS 315: Module 5 Trees Worksheet**

1. Draw the changes to the tree given the following insert operations. (assume key and info are the same)

|  |
| --- |
| 1. tree.put(50, val); |

|  |  |
| --- | --- |
| b. tree.put(32, val); |  |

1. Given the following binary search tree, bst, perform the following operations on it.



* 1. [5 points] Redraw tree after bst.put(30);
  2. [10 points] Using the solution from part a, redraw the tree after bst.delete(40)

1. Given the following binary search trees, **indicate the balance factor for each node (write it beside each node)**. Then, **indicate if the tree is balanced.**

|  |  |
| --- | --- |
| * 1. [8 points] | Balanced: YES / NO |

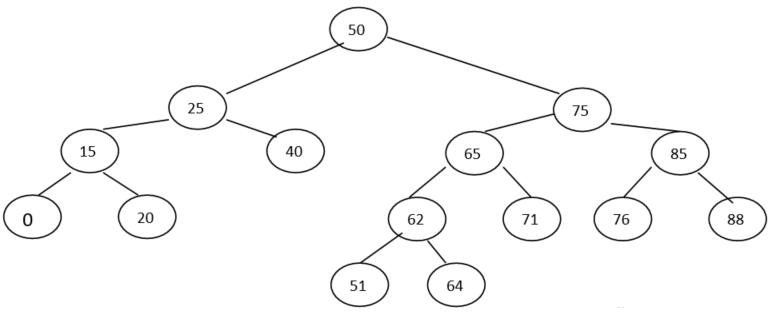
|  |  |
| --- | --- |
| * 1. [8 points] | Balanced: YES / NO |

|  |  |
| --- | --- |
| c. [8 points] | Balanced: YES / NO |

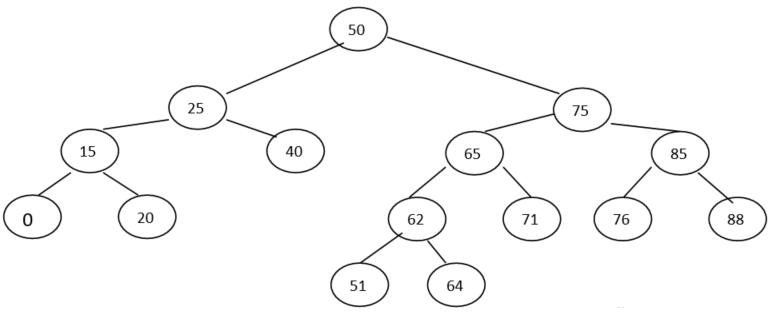
1. An array of elements is added to a binary search tree on element at a time (from left to right).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **W** | **F** | **Z** | **O** | **B** | **G** |
| **0** | **1** | **2** | **3** | **4** | **5** |

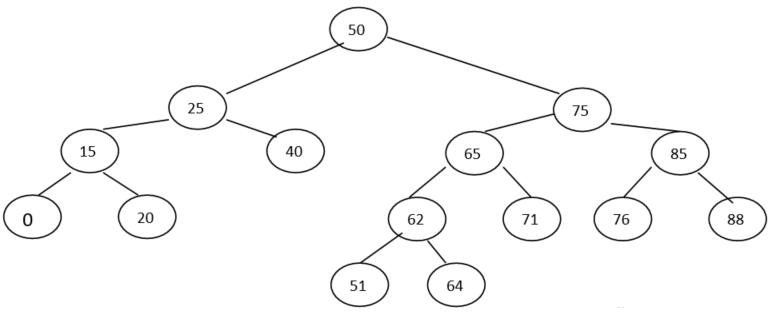
1. Show the binary search tree resulting from incrementally inserting each array element (you are only required to show final resulting tree
2. For your final tree, **label the balance factor for each** node in the tree.
3. Is your tree balanced? **Yes or No**
4. **Given the following tree, show the results of the operations performed on the original tree for each subproblem.**
5. Draw the changes to the tree after tree.delete(20)



1. Draw the changes to the tree after tree.delete(75)



1. Draw the changes to the tree after tree.delete(50)



1. Given the following list perform the requested traversal writing out the output if “visit” printed each location’s key.
2. In-Order Traversal
3. Pre-Order Traversal
4. Post-Order Traversal
5. Breadth First Traversal