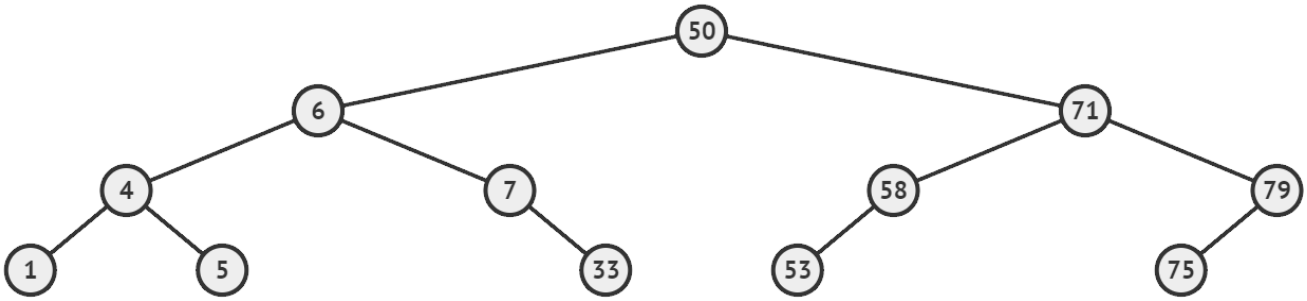
Practice Questions

In order to represent BinaryTrees in a space efficient manner, we may ask you for the values of the nodes in order for a given traversal (pre-order, post-order, breadth-first search). For example given the following tree, if you were asked for the values from a post-order traversal of this tree, then the correct output would be:

1, 5, 4, 33, 7, 6, 53, 58, 75, 79, 71, 50



If you were asked for the values from a pre-order traversal:

50, 6, 4, 1, 5, 7, 33, 71, 58, 53, 79, 75

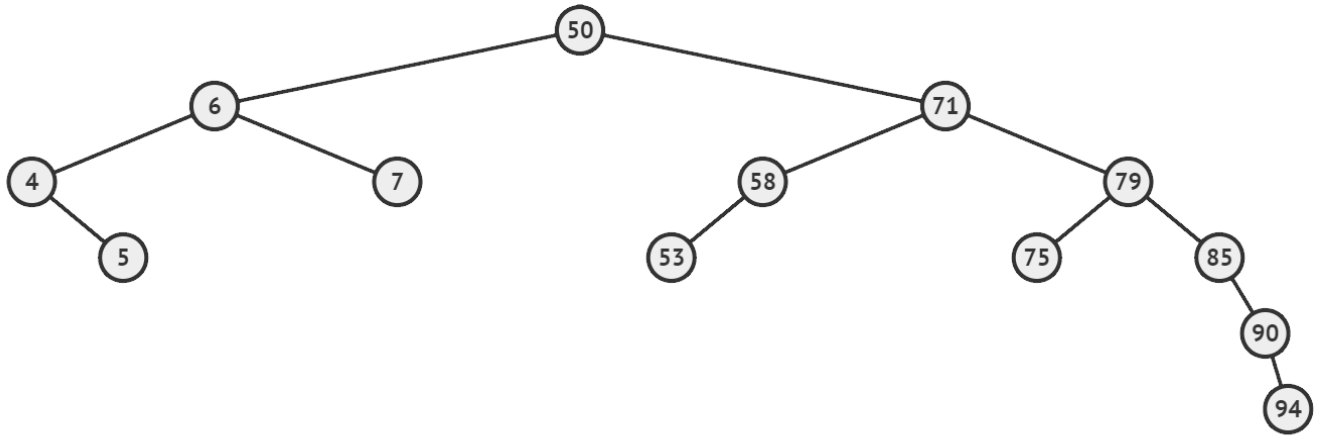
If you were asked for the values from a breadth-first traversal:

50, 6, 71, 4, 7, 58, 79, 1, 5, 33, 53, 75

1. Draw the instance of a BinarySearchTree (as it has been discussed in class) that would be created as a result of the following insertion operations (starting from an empty tree). Once you have done constructing this tree, write all the values on a single line separated by spaces in the order that you would process the nodes using a ***postorder traversal.***

```
add(12), add(62), add(82), add(29), add(6), add(50),
         add(84), add(96), add(42), add(52)
```

2. Consider the following instance of a ScapegoatTree (as it has been discussed in class) where the current overestimate q associated with this Scapegoat tree is 13. Insert 3, 96, and 98 into this Scapegoat tree, then answer the following questions:



a) Which value added triggered the rebuild?

b) how it was decided that this ScapegoatTree must complete a partial rebuild,
(i.e., present and evaluate the calculation required)

c) the location of the vertex selected to be the scapegoat, and
(i.e., present and evaluate the series of calculations required)

d) Output the ***rebuilt subtree*** of this ScapegoatTree using a ***preorder traversal***. DO NOT OUTPUT THE ENTIRE TREE, ONLY THE PART THAT WAS REBUILT. As before, all values go on the same line separated by spaces.

DO BRING A CALCULATOR TO THIS QUIZ (but a copy of this table will be supplied). The table below is provided as a study guide and contains the evaluation of $\log_{3/2} x$ for all values of x between 1 and 20. You will need to evaluate one such value to answer part (a).

$\log_{3/2} 1 = 0.00$	$\log_{3/2} 6 = 4.42$	$\log_{3/2} 11 = 5.91$	$\log_{3/2} 16 = 6.84$
$\log_{3/2} 2 = 1.71$	$\log_{3/2} 7 = 4.80$	$\log_{3/2} 12 = 6.13$	$\log_{3/2} 17 = 6.99$
$\log_{3/2} 3 = 2.71$	$\log_{3/2} 8 = 5.13$	$\log_{3/2} 13 = 6.33$	$\log_{3/2} 18 = 7.13$
$\log_{3/2} 4 = 3.42$	$\log_{3/2} 9 = 5.42$	$\log_{3/2} 14 = 6.51$	$\log_{3/2} 19 = 7.26$
$\log_{3/2} 5 = 3.97$	$\log_{3/2} 10 = 5.68$	$\log_{3/2} 15 = 6.68$	$\log_{3/2} 20 = 7.39$

3. In this first question you will apply the specified operations to instances of LinearHashTable (i.e., hash tables that use the "linear probing" approach as it was discussed in class). When such a hash table is first created, it has $dimension$ 1 and an initial length of 2, and subsequent operations might require that the underlying array be resized, so for each of the questions below, start with the hash table specified and demonstrate (by drawing the underlying array) what the hash table looks like after each operation.

   Since you will only be inserting integer values into these hash tables, the hash function you will use for this question is simply: $hash(x) = x \,\%\, 2^{dimension}$ where the length of the hash table is $2^{dimension}$. To clarify, if the hash table is of length 8, the $dimension$ is 3, so if you apply the $hash$ function to some number, you obtain the result by long dividing that number by 8, keeping only the remainder and discarding the quotient.

   Be sure to enter the proper number of entries separated by spaces. For example, a dimension 2 hash table has 4 locations. Thus I would enter exactly 4 values separated by spaces. For example: N N 100 200

   if my hashtable was storing null in index 0 and 1 and 100 in index 2 and 200 in index 3.

   a) Starting from an empty hash table (of $dimension$ 1), perform the specified insertion operations. Write null values as N. The first one is done for you. If you resize the table, be sure to provide the correct number of N values and in the proper order.

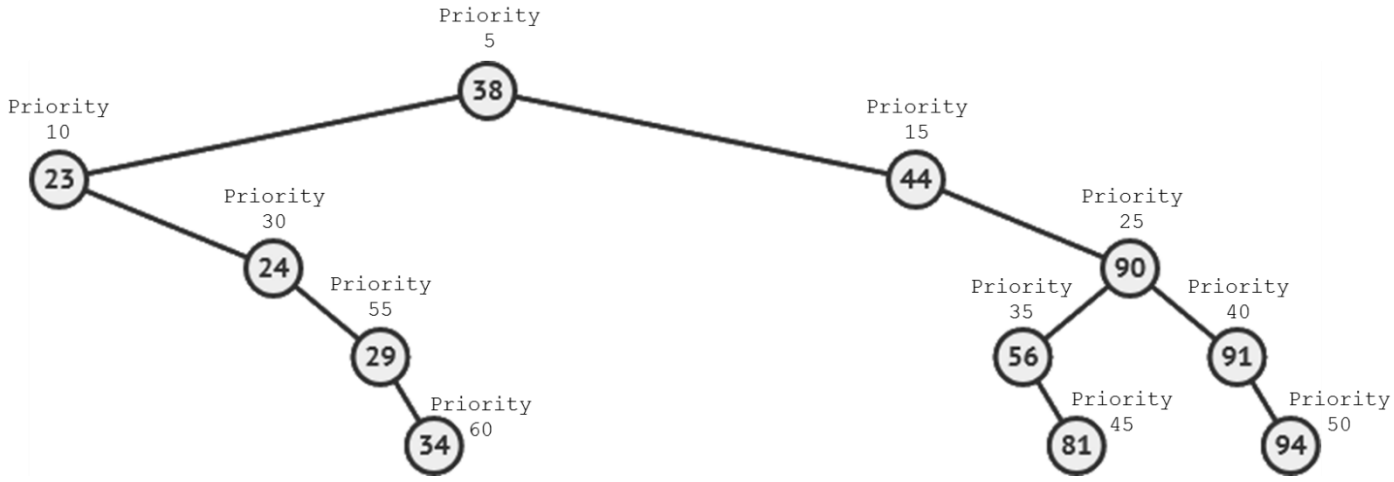
`add(39)`	N 39
`add(63)`	
`add(15)`	
`add(47)`	

b) Starting from the hash table provided (of $dimension$ 3), perform the specified operations.

	N N N N 44 61 30 N
`remove(61)`	
`add(84)`	
`remove(30)`	
`add(37)`	

4. Consider the following instance of the Binary Treap (as it has been discussed in class). The insertion of key value 60 into this treap with a priority 20 would be performed such that the binary search tree property was maintained, but then this insertion might be followed by one or more rotations of the treap in order to maintain the treap property.

   For this question, you must perform this insertion, redrawing the tree after the insertion operation and after each of the required rotation operations. Additional space has been left for this question on the following page. You must show all your work - answers presented without sufficient justification will be penalized.



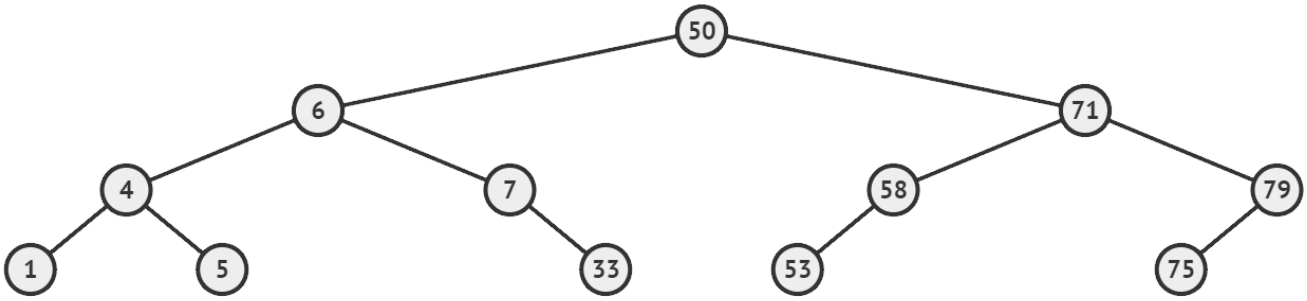
4. (Continued)

5. Consider the following instance of the Binary Heap (as it has been discussed in class), presented here as an array. For this question, you must first draw this heap as a binary tree. Once you have drawn the complete heap, insert the value 8. It will not be in the correct position following its insertion, so it must then be "bubbled up" to maintain the heap property. Specify all the comparisons that must be made (to determine to what position it will "bubble") and then redraw the entire heap in its final form.

Additional space has been left for this question on the following page. You must show all your work - answers presented without sufficient justification will be penalized.

```
[2, 3, 44, 16, 24, 85, 58, 30, 38, 68, 42, 99,
94, 91, 65, 96, 36, 51]
```

6. Consider the following instance of a BinarySearchTree (as it has been discussed in class) and apply the following removal operations. You must apply the removal operations in the order they are listed below. Recall there are two equally valid remove techniques. On this exam you are expected to use the method which uses the *successor node* as the replacement node. When you have applied all the remove operations, output the tree using a *breadth-first traversal.* (As before, all values on a single line separated by spaces.)



a. `remove(50)`
b. `remove(6)`
c. `remove(7)`

7. Recall that an in-order traversal of a BinarySearchTree will visit all the nodes in sorted (based on the sorted order of the data). Given the following Node and method definition for a BinarySearchTree, write a method that returns the Node that **follows** the Node w in an in-order traversal of the tree, *assuming that w.right is nil (n.b.*, this implies that the Node that follows w must be an *ancestor* of w, not a descendant).

```
public class Node<T extends Comparable<T>>{
    T data;
    Node left, right, parent;
}

/**
 * Find the node that follows w in an in-order traversal.
 * Assume that w.right == nil
 * @return the node that follows in an in-order traversal.
 */
public Node findNext(Node w){









}
```