R17.9

Consider the following tree. In which order are the nodes printed by the Binary-SearchTree.print method? The numbers identify the nodes. The data stored in the nodes is not shown.

pre-order: 1->2->4->7->3->5->8->6->9->10

in-order: 2->7->4->1->8->5->3->9->6->10

post-order: 7->4->2->8->5->9->10->6->3->1

In textbook, this method choose in-order one.

```
/**
   Prints a node and all of its descendants in sorted order.
   @param parent the root of the subtree to print
*/
private static void print(Node parent)
{
   if (parent == null) { return; }
   print(parent.left);
   System.out.print(parent.data + " ");
   print(parent.right);
}
```

Meaning it will print 2->7->4->1->8->5->3->9->6->10.

Prove that in any subtree of a max-heap, the root of the subtree contains the largest value occurring anywhere in that subtree. (5 points)

Considering we are building a max-heap, first we arbitrarily build a heap, then we fix that bottom up, right to left. Wherever we have children node greater than its parent node, we promote parent node's largest child, swap parent node and that child node. Then bottom up, meaning the most top node, root will have the largest value of whole heap.

R17.25

Will preorder, inorder, or postorder traversal print a heap in sorted order? Why or why not?

No, because when we build a heap, there is no specific rule for relation between left child and right child of a parent node. We only guarantee parent node is greater than (smaller than in min-heap) its children node in max –heap, meaning left child could be greater or smaller than right child (Non-deterministic). So when we use pre-order, for instance, it will print left child first, but left child is not guarantee to be smaller (or greater) than its parent node and right sibling. So is in-order and post-order, all of them will not print a heap in sorted order.

AmeriDelUnited Airlines wants to give upgrade vouchers to their top log n frequent flyers, based on the number of miles accumulated, where n is the total number of frequent

flyers. Their current algorithm sorts the flyers by the number of miles flown and then scans the sorted list to pick the top log n flyers. This algorithm runs in O(n log n) time. Is it possible to do this in O(n) time? If yes, explain how. If not, explain why not. (Note: You cannot use any forms of counting sort, radix sort, etc.) (5 points)

We can use heap sort, first we build a max-heap arbitrarily, which takes O(n), then we fix that, enforcing it in correct order, bottom up, left to right, promote the greatest child node, swap that node and its parent node, and recursively. This step takes n/4 * 1 + n/8 * 2 + n/16 * 3 + … =O(n).

So after we build a sorted max-heap in O(2n). Then we want to top log n flyers, we just remove log n times, we will get top log n nodes in order, since the top most one is the greatest one, after we remove, we re-sort the heap, keep the root the maximum node. Since remove a node and re-sort will take log n, in this step, we get the top log n will take (log(n) -1) * log(n), the last time we do not need to resort.

In total, we get O(2n+(log(n) -1) * log(n)) —> O(n).