Name: ______________________ Section/Period: __________

Page	Points	Score
2	12	
3	34	
4	20	
5	16	
6	10	
7	14	
8	7	
Total:	113	

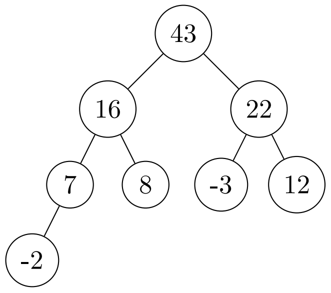
Please copy and sign the following:
"The Naval Service I am a part of is bound by honor and integrity. I will not compromise our values by giving or receiving unauthorized help on this exam."

1. Fill in the following table with the correct **WORST-CASE** run times for each method of each ADT, as implemented with each data structure *except Hashtable*. For Hashtable use **EXPECTED-CASE** for `get()` and `set(k,v)`. Assume everything is implemented the best way we know how. Be sure to clearly mark runtimes arrived at via amortized analysis. [12 pts]

   The hashtable is implemented with **OPEN ADDRESSING**.

		Unsort Arr.	Sor. Arr.	Unsort LL	Sor. LL	Balanced BST	Hashtable
ORDERED MAP	get(k)						
	set(k,v)						
	first()						
	next(k)						

2. Given the below heap, draw the heap after each of two calls to `removeMax()`. What were the two values returned? [18 pts]



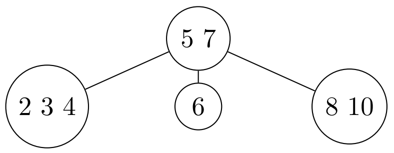
3. Given the following heap in array format, draw the resultant heap after an `insert(55)` and after a `removeMax()`. The X indicates index 0 is empty. [16 pts]
`[X 61 54 57 50 20 39 6 7]`

4. Draw the array that results from adding the following integer priorities in the following order to a heap: `7, 8, 6, 9, 14`. Note this is NOT heapify; each insert should be performed separately. If I can follow your work, step by step, it is much easier to give partial credit. [10 pts]

5. Draw an example of a BST where one node has a balance of 2, and its right child has a balance of -1. Label each node with its sub-tree height and its balance. Use as few nodes as you can. [10 pts]

6. In the following space, draw the AVL tree that results from adding the following integer keys in the following order: `1, 2, 6, 3, 5, 4, 0`. Redraw the entire tree after each rotation, at minimum. [16 pts]

7. Given the following 2-3-4 tree, draw the 2-3-4 tree that results from adding the following integer keys in the following order: `9, 1, 11`. If I can follow your work, step by step, it is much easier to give partial credit. [10 pts]



8. Describe a circumstance in which `insert(k,e)` in a Hashtable would run very slowly. Be sure to include which collison handling method is being used. [3 pts]

9. Consider a hashtable of size 5, implemented with separate chaining. The following strings are inserted, in the following order: `apple` (hash value 2), `banana` (hash value 1), and `cherry` (hash value 1). Draw the resulting hashtable. [6 pts]

10. Consider question 9, but now the hashtable is implemented using open addressing with linear probing. Draw the resulting hashtable. [4 pts]

11. If n elements are stored in a Hashtable of size k, which of the following is the Hashtable's load factor? [1 pts]

    a. $\frac{n}{k}$

    b. $\frac{k}{n}$

    c. $n - k$

    d. $k - n$

12. What is the runtime of heapifying n elements? [1 pts]

    a. $O(\log(n))$

    b. $O(n)$

    c. $O(n \log(n))$

    d. $O(n^2)$

13. Heapify the following array: `[X 7 1 2 8 5 6 3 9 4]`, showing your work. The `X` indicates index 0 is empty. Show the array after each bubble down. [6 pts]