CSCI 4041, Fall 2018, Quiz 2 (30 minutes, 20 points)

Name:

x500:

Discussion Start Time (**circle one**): 3:35 4:40 5:45 6:50 7:45 other:

---

1. (2 points each) True/False - Circle one. Note that when asking about the properties of an algorithm, we specifically mean the version of that algorithm discussed in lecture.

   True False Heap-Extract-Max has a Θ(lg n) worst-case runtime

<In the worst case, a number of exchanges equal to the height of the heap will be needed to restore the heap property, which means Θ(lg n) exchanges>

   True False For extremely large arrays initially in random order, Heapsort will run faster than Insertion Sort.

<Heapsort is Theta(n lg n), Insertion Sort is Theta(n^2)>

   True False Quicksort is in place.

<Quicksort only uses Θ(1) extra memory beyond its input as extra storage space for elements in the original array, so it is in place>

   True **False** If the median element in the subarray is chosen as the pivot for every Partition during Quicksort, the asymptotic runtime will be $\Theta(n^2)$.

<Taking the median as a partition every time divides the problem into two subproblems of size n/2, just like merge sort. So just like merge sort, this results in Θ(n lg n) behavior>

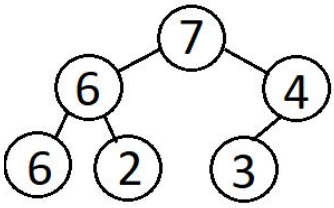
   True **False** All comparison sorts have a Θ(n lg n) worst case runtime.

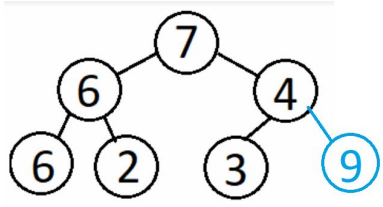
<While all comparison sorts have Ω(n lg n) worst case runtimes, some are strictly worse than n lg n rather than a tight bound, such as Insertion Sort's $\Theta(n^2)$ runtime>

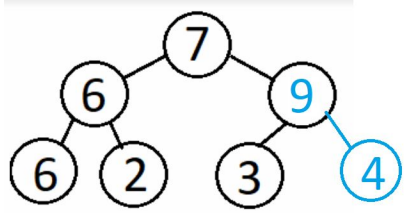
2. (3 points) Give the array that results when Partition(A,3,7) is called on A = [19, 21, **18, 22, 9, 22, 15**, 18] (assume array indexing starting at 1). Showing your work may be useful for awarding partial credit, but is not required.

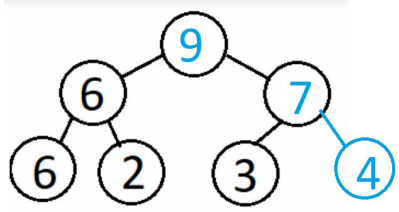
Pivot: A[7] = 15
Swap 9 and 18: A = [19, 21, **9, 22, 18, 22, 15**, 18]
Final Output, swap 22 with pivot 15: A = [19, 21, **9, 15, 18, 22, 22**, 18]

3. (3 points) Give the max heap that results (in tree form, not array form) after Heap-Insert-Key is called on the max heap below, with key 9. Showing your work may be useful for awarding partial credit, but is not required.



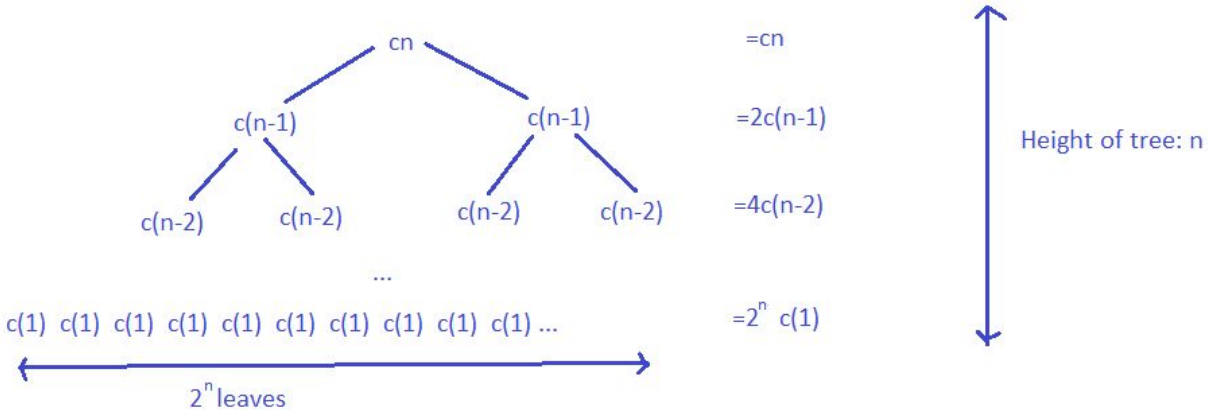






4. (4 points) Suppose that we modified Merge-Sort as shown below, without changing what the subroutine Merge does. Give a tight bound for the worst-case asymptotic runtime of this algorithm, and briefly explain your answer.

```
Merge_Sort(A,p,r)
      if p < r
            Merge_Sort(A, p, r-1)
            Merge_Sort(A, p+1, r)
            Merge(A, p, p, r)
```



Total runtime = $cn + 2c(n-1) + 4c(n-2) + 8c(n-3) + \ldots + 2^n c(1) = \text{Theta}(2^n)$

(2^n term dominates all polynomial factors)

The runtime is $\Theta(2^n)$, because the new Merge-Sort splits into two problems of size (n-1) at each recursion, meaning that the number of Merge-Sort calls double at each recursion level, but unlike normal Merge-Sort, this version will require n recursions to reach the base case. Thus, the final recursion level will involve 2^n calls to Merge-Sort, each of which will take constant time to run. This will dominate the runtime of the levels above, making the final runtime $\Theta(2^n)$.

(This is a bit more explanation than is necessary for the problem: you don't need to draw a tree diagram)