3. **Heap (35 pts;6+6+10+12)**

Consider k max heaps, with n elements in each. We wish to combine these into a single heap, using different approaches in (a) thru (c). For each of these approaches, use the fastest way to execute it, when analyzing the running time. **Show your derivation** (no credit otherwise).

a) We create a new, empty heap H. From each of the k heaps, h, we repeatedly call the *delete* method to remove the highest priority element, and insert it into H, until h is empty. What is the worst-case big O running time? Ignore time to create the initial empty heap H.

b) We create an array A of length kn. For each of the k heaps, we iterate over its internal array, and append its items one at a time to A. When all heaps are emptied out, we convert A into a heap. What is the worst-case big O running time? Ignore time to create the array A.

c) We group the k heaps into $k/2$ pairs, and apply the algorithm from part (b) on each pair, leaving $k/2$ heaps. We then repeat this process until we are left with a single heap. What is the worst-case big O running time?

d) Complete the `siftUp` method in the following `MinHeap` class.

```
public class MinHeap {
    // sifts up in a min heap array starting at index k
    private static void siftUp(int[] minHeap, int k) {
```

4. **Sorting (35 pts;7+9+9+10)**

a) Mergesort is used to sort the items a,b,c,d,e. Come up with an input sequence of these items for which mergesort will use up the most number of comparisons to sort. Show the mergesort tree that results on recursively splitting the input list and sub lists.

b) Quicksort SPLIT is applied to the integer array [4,3,5,7,9,2,1], using the first entry as the pivot. Show every step of the split process where there is a change in the sequence of items. (You are required to show the SPLIT for the initial array ONLY.)

c) What is the worst case big O running time of quicksort? Clearly specify what unit time operations you are counting towards the running time, and show your derivation on a generalized worst case scenario. (Just writing the answer gets no credit.)

d) Suppose you are given traffic data (number of page hits) for n websites for 2016. (n is a large number, in the order of hundreds of thousands.) You are asked to print the top k websites with greatest amount of traffic (k is a small number, not more than 100). Which of insertion sort, mergesort, quicksort, or heapsort would be the fastest (comparing worst cases) for this task? What would be the worst case big O running time, in terms of n and k?

(Note: You cannot modify the steps in any of these known sorting algorithms, but you can stop it as soon as you get the top k. And you can use the running times of all involved known entities without derivation.)

5. Applications (35 pts;25+10)

a) Suppose a hash table is used to count the number of occurrences of each word (case INsensitive) in a text file. Assume the file consists of T words in all, with D distinct words, and T is significantly greater than D.

Derive the worst case big O running time, AND the expected big O running time (i.e. assuming uniform distribution of entries over the hash table locations), to compute the word counts. The time needed to read the words from file should be counted in your result: assume unit time to read each word from the input file. Clearly show all the steps of your derivation.

b) An array holds names of movies, with no duplicates, *sorted in alphabetical order.* Searches are done on the array to find completion lists (matches) given prefixes (starting letters) of movie names. For instance, prefix "Si" would match "Sicario" and "Sin City". The prefix can fully match a movie name as well, but the movie name cannot be smaller than the prefix.

Describe the FASTEST algorithm (no code) for prefix search, by listing the sequence of steps, clearly and concisely. What would be the worst case big O running time of your algorithm if the array had n movie names, and the completion list had k matches?