Part A: Update all java file names to include your initials at the end, send individual .java file only.

1. Write merge method for mergeSort method, it takes the array of data, starting index of first half, starting index of second half and end index of second half. You may test it in this Sort.
2. Write MinheapPriorityQueue constructor, which takes an array of data, and construct the min heap priority queue using bottom-up algorithm. The expected run time should be O(n), where n is the total number of data. BubbleDown method is provided. You may test it in this minHeap.
3. Write a program which takes thee list of names, and print out all the names only once. The expected runtime should be O(n) where n is the total number of names in the lists.
   You don't have to sort the list. Use appropriate data structure to store the names, so add all data to the collection only once efficiently. The read-in list method and list1, list2, list3 is given.
   You may assume that the expected runtime of searching or inserting into a hash table is O(1).
   Do not call the list's contains method, because it slows down the run time.

Part B

1. Show how an initially empty max heap looks like after inserting following elements in the given order. Draw the tree.

   5, 2, 7, 8, 6, 1, 7

2. Show how a bottom-up max heap construction looks like from the above given values. Draw the tree.

3. What is the best-case and worst-case of insertionSort?

4. What is the best-case and worst-case of mergeSort?

5. What is the best-case and worst-case of quickSort?

6. What can we do to lower, if not eliminate completely, the chances of worst-case of quickSort?

Extra Credi:

If we can access the data of a min-heap, how do we find the fourth minimum value efficiently? That means instead of O(log n), can we speed up to an expected constant time?