CS61B Week 10: Hashing and Sorting

1. Complete the following method to agree with its comment.

```
/** This method parses numbers separated by whitespace
 *  from stdin and returns them as a list of doubles.
 *  Ignore comments (everything after a # on the line). */
public static List<Double> parseInput() {
    Scanner s = new Scanner(System.in);
    ArrayList<Double> doubles = new ArrayList<Double>();
```

2. Show the steps taken by quicksort on the following unordered list, assuming that the pivot node is always the rst item in the (sub)list being sorted, and the array is sorted in place. At every step, circle all nodes that will be pivots on the next step, and box all previous pivots.

```
36  22  15  56  48  90  72  06
```

3. Show the steps taken by mergesort on the following unordered list. Show how the list is broken up at every step.

```
36  22  15  56  48  90  72  06
```

4. Show the steps taken by LSD radix sort on the following unordered list. Show the different buckets at every step.

```
102  351  232  451  998  754  325  425  443  564  987  882  672  289
```

5. Fill out the following implementation of a HashSet.

```
import java.util.LinkedList;
import java.util.ArrayList;
public class HashSet<T> {
    int numBuckets;
    ArrayList<LinkedList<T>> buckets;
    float loadFactor;
    int numKeys;
```

```
    public HashSet(float lf) {
        numBuckets = 10;
        buckets = new ArrayList<LinkedList<T>>(numBuckets);
        for (int i = 0; i < numBuckets; i++)
            buckets.add(i, new LinkedList<T>());
        loadFactor = lf;
        numKeys = 0;
    }
    /** Adds the given OBJECT of type T to your HashSet.
     *  If numKeys / numBuckets > loadFactor then double your numBuckets
     *  and the size of your keys array. Resize is implemented for you. */
    public void add(T object) {









    }
    /** Resizes the HashSet to have NEWSIZE number of buckets. */
    public void resize(int newSize) {
        ArrayList<LinkedList<T>> oldBuckets = buckets;
        buckets = new ArrayList<LinkedList<T>>(newSize);
        for (int i = 0; i < newSize; i++)
            buckets.add(i, new LinkedList<T>());
        for (LinkedList<T> bucket : oldBuckets)
            for (T key : bucket)
                add(key);
    }
    /** Returns true if the HashSet contains OBJECT, false otherwise. */
    public boolean contains(T object) {









    }
    /** Deletes OBJECT from the HashSet, if it is in the set. */
    public void delete(T object) {









    }
}
```

Sample Interview Question of the Week:
Given an array of n unsorted integers, what is the most efficient way to find the kth smallest element of the array? Can you do it in less than $\Theta(log(n))$ time? Can you do it in average $\Theta(n)$ time?