CSE 373 Practice Problems, Round 1

1) Given a value ‘x’ and an array of integers, determine whether two of the numbers add up to ‘x’:

```
boolean x_sum(input_array, x):
     sort(input_array)
     left = 0
     right = input_array.length - 1
     while left < right:
          if input_array[left] + input_array[right] == x:
               return true
          else if input_array[left] + input_array[right] < x:
               left++
          else
               right--

     return false
```

2) Given an array of integers, return a new array of the same values without any duplicates

```
int[] no_duplicates(input_array):
     Set s = new Set
     for (int x : input_array):
          s.add(x)
     return s.toArray()
```

3) Given an array that contains the values 1 through ‘n’ two times each, find the one number that is contained only 1 time.

```
int missing(input_array):
     HashMap m = new HashMap
     for (int x : input_array)
          if !m.containsKeu(x)
               m.put(x, 1)
          else
               m.put(x, m.get(x) + 1)

     for (int x : m.keys)
          if m.get(x) == 1
               return x
```

4) Given a list of integers, find the highest value obtainable by concatenating them together.
For example: given [9, 918, 917], result = 9918917
For example: given [1, 112, 113], result = 1131121

```java
import java.util.*;
public class IntConcat {
    private static Comparator<Integer> sorter = new Comparator<Integer>(){
        @Override
        public int compare(Integer o1, Integer o2){
            String o1s = o1.toString();
            String o2s = o2.toString();
            if(o1s.length() == o2s.length()){
                return o2s.compareTo(o1s);
            }
            int mlen = Math.max(o1s.length(), o2s.length());
            while(o1s.length() < mlen * 2) o1s += o1s;
            while(o2s.length() < mlen * 2) o2s += o2s;

            return o2s.compareTo(o1s);
        }
    };

    public static String join(List<?> things){
        String output = "";
        for(Object obj:things){
            output += obj;
        }
        return output;
    }

    public static void main(String[] args){
        List<Integer> ints1 = new ArrayList<Integer>(Arrays.asList(1, 34, 3, 98,
9, 76, 45, 4));
        Collections.sort(ints1, sorter);
        System.out.println(join(ints1));
        List<Integer> ints2 = new ArrayList<Integer>(Arrays.asList(54, 546, 548,
60));

        Collections.sort(ints2, sorter);
        System.out.println(join(ints2));
    }
}
```

5) Given a very large file of integers (more than you can store in memory), return a list of the largest 100 numbers in the file

```
def getLargest(input)
     minHeap mh = new minHeap
     put first 100 values from input into minHeap
     while input.hasNext()
          next = input.next
          if input.getLargest < next
               mh.removeMin
               mh.insert(next)
          print mh
```

6) Given a list of strings, write a method that returns the frequency of the word with the highest frequency.

```
def returnHighest(input_array)
     HashMap m = new HashMap
     max = 0
     for (String x: input_array)
          if !m.containsKeu(x)
               m.put(x, 1)
          else
               curr = m.get(x) + 1
               m.put(x, curr)
     return curr
```

7) Given a list of strings, write a method that returns a sorted list of words based on frequency

```
def returnHighest(input_array)
     HashMap m = new HashMap
     max = 0
     for (String x: input_array)
          if !m.containsKeu(x)
               m.put(x, 1)
          else
               curr = m.get(x) + 1
               m.put(x, curr)
     add each value to priorityqueue
     output priorityqueue traversal
```

8) Given an array of strings, where each string is sorted lexicographically, determine the order of characters in the given alphabet.
For example, given the english alphabet, the ordering is: “a,b,c,d,e,f . . . x,y,x”.

Your output should be the lexicographic order of only the characters that were found in the input strings.

For example: input = [xyz, yk, zk, xm, my], then the output would be [x,m,y,z,k]

```
def build_ordering(input_array)
     for (String s: input_array)
          for each consecutive chars:
               if chars hasn't been seen:
                    add node to graph
               add edge in graph between
     run kruscals algorithsm for topological sort
```