1.Recursive Concepts – 15 points (5+5+5)

What is the program output of `mysteryA(7)` ? `7,4,1,1,4,2,2,7,`

Circle the correct response: `mysteryA` is an example of a **TREE** of activations.

(1) Which line in the above code implements a recursive case? **2**

(2) Decide if each statement is true or false for `mysteryB`? Circle (or write in) the correct response.

A	It is tail recursive.	True / **False**
B	It is an example of a chain of activations.	True / **False**
C	It is an example of an infinite *loop* when n is negative.	True / **False** (∞ **R.)**
D	The variable `s` is shared over all activations.	True / **False**

Complete the following recursive code that counts the number of digits of an integer that are 6 or greater. For example, `count(471629034)` returns 3. Hint `n%10` and `n/10` may be useful.

```
public static int count(int n){
   if(n<6) return 0;
   if(n<10) return 1; //6,7,8,9
   return count(n/10) + count(n%10);
}
```

2.Tracing Code – 15 points (3+2+10)

```
1    public static int foo(int a, int b){
2       if (a == b) return b;
3       if (a > b) return 2 * foo(a - 2,b - 1);
4       return a + foo(a + 9,b + 1);
5    }
```

a. (i) Carefully explain why `foo` is an example of forward recursion. (ii) Underline or circle the relevant mathematical operators in `foo` that support your answer.
Line3 and Line 4: Addition and multiplication operations (underlined) after recursive call completes. Note underlining/circling "a+9,b+1" or "a-2,b-1" is INCORRECT.

b. Which one of the following statements is true for the execution of `foo(3,3)`? __**E**____

c. Create an activation diagram below for the execution of `foo(0,5)`. For full marks ensure your activation diagram includes:

A. The method parameter values for each execution of `foo`.

B. Label the return arcs with the returned value, including the returned value of `foo(0,5)`.

C. Use your diagram to determine the returned value of `foo(0,5)` and write it here: **24**

D. How many times is foo activated (called), including the first `foo(0,5)` ? **5**

5 activations (return values not shown):(0 5) - (9 6) - (7 5) - (5 4) - (3 3)

3. Linked Lists – 15 points (5+5+5)

```
public Link insert(int v) {
        if( v < this.value) return new Link( v,this );
        if( next != null) next = next.insert(v);
        else next = new Link( v,null);
        return this ; //we don't need to move.
     }
public int sub2(int acc) {
   acc++;
   value -= 2;
   if(acc <4 && next != null) return value + next.sub2(acc);
   return value;
}
public Link check() {
   if(next == null) return null;
   if(value > next.value) return this;
    return next.check();
}
```

4. The Recursive Car Mechanic – 15 points (8 + 7)

```
int max(){
            int m = 0;
            if(yes != null) m  = yes.max();
            if (no != null) m = Math.max(m,no.max());
           return 1 + m;
         }
public Question find() {
  if((yes == null || no == null ) && question.indexOf("door") != -1) return this;
  if(yes!=null) {Question q = yes.find(); if(q!=null) return q;}
  if(no != null) return no.find();
 return null;
}
```

5. The SMS Explainer with a Binary Search Twist – 15 points (10 + 5)

```
public class Lookup {
  public static Pair search(Pair[] data, String key, int lo, int hi){
    if(lo>hi) return null;
    int mid = (lo + hi)/2; // OK if lo,hi not too big

    String s = data[mid].sms;
    if(s.equals(key) return data[mid];
    if(s.compareTo(key)<0) return search(data,key,mid+1,hi);
    return search(data,key,lo,mid-1);
}
public static String toPhrase(Pair[] data, String key) {
   Pair p = search(data,key,0, data.length-1);
   if(p != null) return p.phrase;
   return "?";
 }
```

6. Recursion? See Question 6! - 15 points (3+3+3+3+3)

```
   public static int count(int[] data, int max,  int lo, int hi) {
      if(lo>hi) return 0; // Overshot! (can return 0 because lo != hi)
      int c = count(data,max,lo+1,hi);
      if(data[lo] <= max) return 1 + c;
      return c;
   }
```

b. Using a binary search each time you make a comparison against the key.... **N = 7**

c. "A simple linear search of all values stored... **is not sorted by the key value**"

d. **//** **"Fails when the min value is only at the last link."**

```
Link findLast(Link result) {
  if(value == 0)  result = this
  if(next != null) return next.findLast(result);
  else return result;
}
```

7. Selection Sort – 10 points (3 + 4 + 3)

```
public static void swap(double[] data, int i, int j) {
 double t = data[i];
 data[i] = data[j];
 data[j] = t;
}
public static void sort(double[] data, int lo, int hi) {
 if(lo<hi) { swap(data,lo, findMin(data,lo,hi)); sort(data,lo+1,hi);}
}
public static void selectionSort (double[] data) {
 sort(data,0,data.length -1);
}
}
```