CSE 143 Midterm, August 1, 2011 **Sample Solution**

Question 1. (15 points) ArrayLists. Consider the following method:

```
public static void scramble(ArrayList<String> list) {
   for (int i = list.size()-1; i >= 0; i = i-2) {
      String s = list.get(i);
      list.remove(i);
      list.add(s);
   }
}
```

Now, suppose we have an `ArrayList` named `words` containing the following strings:

```
["eye", "frog", "of", "from", "newt", "toe"]
```

What output is produced if we call method `scramble` with this `ArrayList` as an argument, and then print the list after the method returns, as follows:

```
scramble(words);
System.out.println(words);
```

(Don't worry about the punctuation – the number and order of the words in the output is what's important.)

Output:

```
[eye, of, newt, toe, from, frog]
```

Question 2. (12 points) Complexity. Each of the following methods processes arrays of integers. For each method, circle the complexity ($O(1)$, $O(n)$, etc.) that best characterizes its execution time as a function of the size of its array argument. (i.e,. The array size is the problem size.)

Hint: don't worry too much about what each method does. Just figure out how many steps it takes.

(a)
```
// modify elements of array a in a strange way
void mangle(int[] a) {
   for (int i=0; i < a.length; i++) {
      if (a[i] < 0) {
         a[i] = -a[i];
      } else {
         a[i] = 2*a[i];
      }
   }
}
```

Circle: $O(1)$ (**$O(n)$**) $O(n^2)$ $O(n^3)$ $O(2^n)$

(b)
```
// rearrange elements of a
void mumble(int[] a) {
   int left  = 0;
   int right = a.length-1;
   while (left < right) {
      int temp = a[left];
      a[left]  = a[right];
      a[right] = temp;
      left++;
      right--;
   }
}
```
(Reminder: `n++` increases `n` by 1, `n--` decreases `n` by 1)

Circle: $O(1)$ (**$O(n)$**) $O(n^2)$ $O(n^3)$ $O(2^n)$

Question 3. (20 points) Stacks/queues. Complete the method isBalanced below so that it returns true if its string argument contains properly nested and balanced parentheses ('(' and ')') and square brackets ('[' and ']'), and returns false if not. A string contains properly nested and balanced parentheses and brackets if there is exactly one right parenthesis or bracket for each left parenthesis or bracket, and pairs of parentheses and brackets are properly nested. Other characters in the string are ignored. Examples:

```
isBalanced("(this) is [balanced (properly)], isn't it?")  => true
isBalanced("(this is [not) okay]")          => false  (() and [] pairs don't match)
isBalanced("(this (isn't okay)")            => false  (not enough closing ')'s)
isBalanced("this is okay ([(())][[()]])") => true
```

For full credit you may use only **one** queue or stack in your solution (but not both). You may have as many additional simple variables as you like, but no additional collections, arrays, or strings.

Hint: The logic needed has a close resemblance to that needed for the HTML validator assignment.

```
public static boolean isBalanced(String s) {
   Stack<Character> stack = new Stack<Character>();
   for (int i = 0; i < s.length(); i++) {
      char ch = s.charAt(i);
      if (ch =='(' || ch =='[') {
         stack.push(ch);
      } else if (ch == ')') {
         if (stack.isEmpty() || stack.pop() != '(') {
            return false;
         }
      } else if (ch == ']') {
         if (stack.isEmpty() || stack.pop() != '[') {
            return false;
         }
      }
   }
   return stack.isEmpty();
}
```

Many answers used strings of length 1 instead of characters to process the contents of the string. As long as that was done properly, the answers received full credit.

Question 4. (20 points) Linked lists. Suppose we have a class SortedStringLinkedList that uses a linked list made up of ListNode objects to store a collection of strings in non-decreasing order.

Complete the definition of method removeDuplicates, below, so that when it is called it deletes duplicate values from the sorted list. For example if the list contains

["apple", "apple", "banana", "banana", "cherry", "donut", "donut", "donut", "eggplant"]

then after calling method removeDuplicates the list should contain

["apple", "banana", "cherry", "donut", "eggplant"]

For full credit you must remove duplicate ListNodes from the list, and you may not change the data stored in any ListNode, create any new ListNodes, or use any other collections, strings, or arrays.

Hint: The solution can be quite simple. If you have complicated, nested loops, see if you can simplify.

```
public class SortedStringLinkedList {
   private ListNode list; // list data
   ...
```

```
public class ListNode {
   public String data
   public ListNode next;
   ...
```

```
public void removeDuplicates() {
   ListNode p = list;
   while (p != null){
      // inv: p.data is a unique value to be retained
      if (p.next != null && p.data.equals(p.next.data)) {
         // next node is a duplicate of p, remove it
         p.next = p.next.next;
      } else {
         // next node is different, advance p
         p = p.next;
      }
   }
}
```

Question 5. (18 points) Collections. Suppose we have a collection of type Map<String,Double> that contains a map from names to debit card balances. For example, the map might contain the following key/value pairs:

```
{Bart=-17.42, Marge=35.12, Lisa=1435.21, Homer=-3.55, Krusty=0.00}
```

Complete method printOverspent below so that it prints the name and balance for each name that has a negative debit card balance associated with it. The names may be printed in any order; each separate name and balance pair should appear together on a new output line. Names and balances should not be printed for those names whose balance is greater than or equal to zero. Given the above data, the output should be

```
Bart  -17.42
Homer  -3.55
```

(The same two lines in the reverse order would also be okay.)

You may assume that the Map argument is not null and that no name or balance is null. You should not modify the original list.

```
public void printOverspent(Map<String,Double> accounts) {
   Set<String> names = accounts.keySet();
   for (String name: names) {
      if (accounts.get(name) < 0) {
         System.out.println(name + accounts.get(name));
      }
   }
}
```

Question 6. (15 points) Recursion. Consider the following method.

```
public static int f(int n, int m) {
   if (n == 0) {
      return m;
   } else if (m == 0) {
      return n;
   } else {
      return f(n/10, m/10)*100 + (n%10)*10 + m%10;
   }
}
```

What value is returned by the method call `f(123,864)`?

It would be helpful if you showed your work and partial results so we can follow the sequence of recursive calls and argument values. That should make it easier for you to get the correct answer and easier for us to award partial credit if something goes wrong along the way. Just be sure that you clearly show your final answer at the end.

182634