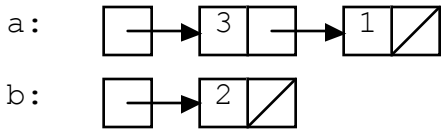
CSE 143 SUMMER 2010 MIDTERM SOLUTION

1. (9 points) Recall the definition of the `ListNode` class:

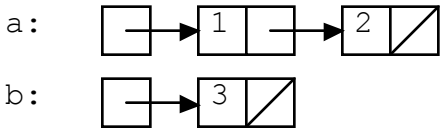
```
public class ListNode {
     int data;
     ListNode next;
}
```

Below are pictures of linked nodes before and after some changes. Write code that will produce the given result. You may use temporary variables. You are NOT allowed to change any `data` fields.

BEFORE:



AFTER:



Four possible solutions:

```
ListNode temp = a;
a = a.next;
a.next = b;
b = temp;
b.next = null;
```

```
ListNode temp = b;
b = a;
a = a.next;
a.next = temp;
b.next = null;
```

```
ListNode temp = a.next;
a.next = null;
temp.next = b;
b = a;
a = temp;
```

```
a.next.next = b;
b = a;
a = a.next;
b.next = null;
```

2. (10 points) Consider the following method:

```
public void mystery(int a, int b) {
     if (a < 0) {
          System.out.println(a);
     } else {
          System.out.print(b + " ");
          mystery(b, a - b);
     }
}
```

For each call below, indicate what output is produced by the method. If the call results in infinite recursion, write out the first 5 characters followed by "…".

Method Call	Output Produced
`mystery(-1, 10)`	-1
`mystery(5, 7)`	7 -2 -2
`mystery(30, 7)`	7 23 -16 -16
`mystery(0, 0)`	0 0 0…
`mystery(10, 10)`	10 0 10 -10 -10

3. (10 points) Write a static method `collapse` that takes an array of integers as a parameter and that returns an array whose elements are the sums of pairs of elements from the array parameter. You may assume that the array passed has an even length.

For example, if an array `nums` has the elements [0, 1, 2, 3, 4, 5], then the call `collapse(nums)` would return an array with the elements [1, 5, 9]. 1 is the sum of 0 and 1. 5 is the sum of 2 and 3. 9 is the sum of 4 and 5.

```
public static int[] collapse(int[] nums) {
    int[] result = new int[nums.length / 2];
    for (int i = 0; i < nums.length; i += 2) {
        result[i/2] = nums[i] + nums[i+1];
    }
    return result;
}
```

```
public static int[] collapse(int[] nums) {
    int[] result = new int[nums.length / 2];
    for (int i = 0; i < result.length; i++) {
        result[i] = nums[2*i] + nums[2*i+1];
    }
    return result;
}
```

4. (5 points) What is the running time of this method? Circle one: **O(n)**

```
public void mystery(int[] array) {
     int margin = Math.min(15, array.length);

     int max = -1;
     for (int i = 0; i < array.length - margin; i++) {
          if (array[i] > max) {
               max = array[i];
          }
     }
     System.out.println("The max is: " + max);
}
```

Explain your answer in 100 words or less.

This method essentially loops through the entire array (except for the last 15 elements) doing a constant amount of work for each iteration (simple if statement). As the array grows larger, the "margin" becomes insignificant compared to the size of the array. Thus, if we say that the size of the array is n, then the running time is O(n).

Student solutions were split between getting the answer right or forgetting to mention how the margin affects the loop (2 point deduction).

5. (15 points) Recall the definition of the `ListNode`:

```
public class ListNode {
     int data;
     ListNode next;
}
```

Recall the `LinkedIntList` class has a single field of type `ListNode` called `front`:

```
public class LinkedIntList {
     private ListNode front;

     <methods>
}
```

Write a method `collapse(void)` for the `LinkedIntList` class that for every pair of nodes, the method replaces the data field of the first node with the sum of the data fields of both nodes. The second node is also removed from the list. If there is an odd number of nodes, the last node will remain in the list.

For example, if the list contained [0,1,2,3,4,5,6], then a call to `collapse()` would change the list to be [1,5,9,6]. 1 is the sum of 0 and 1. 5 is the sum of 2 and 3. 9 is the sum of 4 and 5. 6 is an odd node and is left untouched.

You may NOT create any new nodes.

```
public void collapse() {
  ListNode current = front;
  while (current != null && current.next != null) {
       current.data = current.data + current.next.data;
       current.next = current.next.next;
       current = current.next;
  }
}
```

6. (15 points) Write a method `swap` that takes a `Queue<Integer>` and a `Stack<Integer>` as parameters and that swaps the contents of the queue with that of the stack. For example,

If the queue initially contained:

1	2	3	4

and the stack initially contained:

5
6
7

After the call to `swap`, the queue now contains

5	6	7

and the stack now contains

1
2
3
4

For your convenience, the `Queue` and `Stack` interfaces are on the previous page. You may NOT create any auxiliary data structures.

```
public static void swap(Queue<Integer> q,
                        Stack<Integer> s) {
  int q_size = q.size();

  while (!q.isEmpty()) {
       s.push(q.dequeue());
  }

  while (!s.isEmpty()) {
       q.enqueue(s.pop());
  }

  for (int i = 0; i < q_size; i++) {
       s.push(q.dequeue());
  }
}
```

7. (20 points) Consider the following definitions:

```
public class Seal extends Buster {
     public void method3() {
          System.out.println("Seal 3");
          super.method3();
     }
}

public class Buster extends Bluth {
     public void method2() {
          System.out.println("Buster 2");
     }

     public void method3() {
          System.out.println("Buster 3");
     }
}

public class Bluth {
     public void method1() {
          method3();
          System.out.println("Bluth 1");
     }

     public void method3() {
          System.out.println("Bluth 3");
     }
}

public class Gob extends Bluth {
     public void method1() {
          System.out.println("Gob 1");
     }

     public void method2() {
          System.out.println("Gob 2");
     }
}
```

And assuming the following variables have been defined:

```
Bluth var1 = new Buster();
Object var2 = new Gob();
Buster var3 = new Seal();
Bluth var4 = new Bluth();
Bluth var5 = new Seal();
```

In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in "`a/b/c`" to indicate three lines of output with "`a`" followed by "`b`" followed by "`c`". If the statement causes an error, fill in the right-hand column with either the phrase "compiler error" or "runtime error" to indicate when the error would be detected.

Statement	Output
`var1.method1();`	Buster 3 / Bluth 1
`var1.method3();`	Buster 3
`var2.method1();`	compiler error
`var3.method1();`	Seal 3 / Buster 3 / Bluth 1
`var4.method2();`	compiler error
`var5.method3();`	Seal 3 / Buster 3
`((Bluth)var2).method2();`	compiler error
`((Bluth)var2).method1();`	Gob 1
`((Gob)var4).method2();`	runtime error
`((Bluth)var5).method3();`	Seal 3 / Buster 3

8. (15 points) **Using recursion**, write a method `interleave` that takes two strings as parameters and returns a new string that alternates the characters of the string parameters. For example, the call `interleave("abcdef", "123")` will return `"a1b2c3def"`. You may NOT use any loops (*i.e.*, `for` or `while`).

You are limited to the following String methods:

`charAt(index)`	Returns the character at the given index
`equals(other)`	Returns whether this String is equal to the other object
`length()`	Returns the length of the String
`substring(fromIndex, toIndex)`	Returns a new string that is a substring of this string from startIndex (inclusive) to stopIndex (exclusive)
`substring(fromIndex)`	Returns a new string that is a substring of this string from startIndex (inclusive) to the end of the String

For example, if a String `s` stores the text `"hello"`, then:

[EXAMPLES CUT]

```
public static String interleave(String a, String b) {
   if (a.length() == 0) {
        return b;
   } else {
        return a.charAt(0) +
           interleave(b, a.substring(1));
   }
}

public static String interleave(String a, String b) {
   if (a.length() == 0 || b.length() == 0) {
        return a + b;
   } else
        return a.charAt(0) + b.charAt(0) +
           interleave(a.substring(1), b.substring(1));
   }
}
```