Solution to CSE143X Section #13 Problems 2

```
1.      Method Call                       Output Produced
        -------------------------------------------------
        mystery(2);                       2+
        mystery(5);                       52+-
        mystery(7);                       731---
        mystery(18);                      8942++-+
        mystery(21);                      1052+-+-
```

2. One possible solution appears below.

```java
   public String undouble(String s) {
       if (s.length() < 2) {
           return s;
       } else if (s.charAt(0) == s.charAt(1)) {
           return s.charAt(0) + undouble(s.substring(2));
       } else {
           return s.charAt(0) + undouble(s.substring(1));
       }
   }
```

```
3.      Statement                         Output
        -----------------------------------------------------------
        var1.method2()                    Box 2
        var2.method2()                    Jar 2
        var3.method2()                    Cup 2/Box 2
        var4.method2()                    Jar 2
        var5.method2()                    compiler error
        var6.method2()                    Pill 2
        var1.method3()                    Box 2/Box 3
        var2.method3()                    compiler error
        var3.method3()                    Cup 2/Box 2/Box 3
        var4.method3()                    Jar 2/Box 3
        ((Cup)var1).method1()             runtime error
        ((Jar)var2).method1()             Jar 1
        ((Cup)var3).method1()             Cup 1
        ((Cup)var4).method1()             runtime error
        ((Jar)var4).method2()             Jar 2
        ((Box)var5).method2()             Box 2
        ((Pill)var5).method3()            compiler error
        ((Jar)var2).method3()             Jar 2/Box 3
        ((Cup)var3).method3()             Cup 2/Box 2/Box 3
        ((Cup)var5).method3()             runtime error
```

```
4.      before                 after                    code
------------------------+----------------------+---------------------------
 p->[1]->[2]            | p->[1]->[2]->[3]     | p.next.next = q;
                        |                      | q = null;
 q->[3]                 | q                    |
------------------------+----------------------+---------------------------
 p->[1]                 | p->[2]->[1]          | ListNode temp = q;
                        |                      | q = q.next;
 q->[2]->[3]            | q->[3]               | temp.next = p;
                        |                      | p = temp;
------------------------+----------------------+---------------------------
```

```
4.     before                   after                      code
----------------------+----------------------+----------------------------
 p->[1]->[2]          | p->[2]->[4]          | p.next.next = q.next;
                      |                      | q.next = null;
 q->[3]->[4]          | q->[1]->[3]          | ListNode temp = p;
                      |                      | p = p.next;
                      |                      | temp.next = q;
                      |                      | q = temp;
----------------------+----------------------+----------------------------
 p->[1]               | p->[2]->[1]->[4]     | p.next = q.next.next;
                      |                      | q.next.next = null;
 q->[2]->[3]->[4]->[5]| q->[5]->[3]          | ListNode temp = p;
                      |                      | p = q;
                      |                      | q = temp.next.next;
                      |                      | q.next = p.next;
                      |                      | p.next = temp;
                      |                      | temp.next.next = null;
----------------------+----------------------+----------------------------
```

5. One possible solution appears below.

```java
    public void removeMax() {
        if (size == 0) {
            throw new IllegalStateException();
        }
        int max = 0;
        for (int i = 1; i < size; i++) {
            if (elementData[i] > elementData[max]) {
                max = i;
            }
        }
        for (int i = max; i < size - 1; i++) {
            elementData[i] = elementData[i + 1];
        }
        size--;
    }
```

6. One possible solution appears below.

```java
    public void makePalindrome(Stack<Integer> s) {
        Queue<Integer> q = new LinkedList<>();
        while (!s.isEmpty()) {
            q.add(s.pop());
        }
        for (int i = 0; i < q.size(); i++) {
            int n = q.remove();
            q.add(n);
            s.push(n);
        }
        int oldSize = q.size();
        for (int i = 0; i < oldSize; i++) {
            int n1 = q.remove();
            int n2 = s.pop();
            if (n1 == n2) {
                q.add(n1);
            }
        }
        while (!q.isEmpty()) {
            s.push(q.remove());
        }
    }
```