CSE143 Section #12 Problems

1. Recursive Tracing, 15 points. Consider the following method:

```
public void mystery(int n) {
    System.out.print(n % 10);
    if (n >= 3) {
        mystery(n / 2);
    }

    if (n % 2 == 0) {
        System.out.print("+");
    } else {
        System.out.print("-");
    }
}
```

For each call below, indicate what output is produced:

Method Call	Output Produced
mystery(2);	_____
mystery(5);	_____
mystery(7);	_____
mystery(18);	_____
mystery(21);	_____

2. Recursive Programming, 15 points. Write a recursive method called undouble that takes a string as a parameter and that returns a new string obtained by replacing every pair of repeated adjacent letters with one of that letter. For example, the String "bookkeeper" has three repeated adjacent letters ("oo", "kk", and "ee"), so undouble("bookkeeper") should return the string "bokeper". Below are more sample calls:

Method Call	Value Returned	Method Call	Value Returned
undouble("odegaard")	"odegard"	undouble("oops")	"ops"
undouble("baz")	"baz"	undouble("foobar")	"fobar"
undouble("mississippi")	"misisipi"	undouble("apple")	"aple"
undouble("carry")	"cary"	undouble("berry")	"bery"
undouble("juggle")	"jugle"	undouble("theses")	"theses"
undouble("little")	"litle"	undouble("")	""

You may assume that the string is composed entirely of lowercase letters, as in the examples above, and that no letter appears more than two times in a row. Notice that the method might be passed an empty string, in which case it returns an empty string. You are not allowed to construct any structured objects to solve this problem other than strings (no array, ArrayList, StringBuilder, Scanner, etc) and you may not use a while loop, for loop or do/while loop to solve this problem; you must use recursion. You may use only the string methods included on the cheat sheet.

3. Details of inheritance, 20 points. Assuming that the following classes have been defined:

```
public class Cup extends Box {
    public void method1() {
        System.out.println("Cup 1");
    }

    public void method2() {
        System.out.println("Cup 2");
        super.method2();
    }
}

public class Pill {
    public void method2() {
        System.out.println("Pill 2");
    }
}

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

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

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

    public void method3() {
        method2();
        System.out.println("Box 3");
    }
}
```

And assuming the following variables have been defined:

```
Box var1 = new Box();
Pill var2 = new Jar();
Box var3 = new Cup();
Box var4 = new Jar();
Object var5 = new Box();
Pill var6 = new Pill();
```

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.method2();`	_____
`var2.method2();`	_____
`var3.method2();`	_____
`var4.method2();`	_____
`var5.method2();`	_____
`var6.method2();`	_____
`var1.method3();`	_____
`var2.method3();`	_____
`var3.method3();`	_____
`var4.method3();`	_____
`((Cup)var1).method1();`	_____
`((Jar)var2).method1();`	_____
`((Cup)var3).method1();`	_____
`((Cup)var4).method1();`	_____
`((Jar)var4).method2();`	_____
`((Box)var5).method2();`	_____
`((Pill)var5).method3();`	_____
`((Jar)var2).method3();`	_____
`((Cup)var3).method3();`	_____
`((Cup)var5).method3();`	_____

4. Linked Lists, 15 points. Fill in the "code" column in the following table providing a solution that will turn the "before" picture into the "after" picture by modifying links between the nodes shown. You are not allowed to change any existing node's data field value and you are not allowed to construct any new nodes, but you are allowed to declare and use variables of type ListNode (often called "temp" variables). You are limited to at most two variables of type ListNode for each of the four subproblems below.

   You are writing code for the ListNode class discussed in lecture:

```
public class ListNode {
    public int data;       // data stored in this node
    public ListNode next;  // link to next node in the list

    <constructors>
}
```

   As in the lecture examples, all lists are terminated by null and the variables p and q have the value null when they do not point to anything.

before	after	code
p->[1]->[2] q->[3]	p->[1]->[2]->[3] q	
p->[1] q->[2]->[3]	p->[2]->[1] q->[3]	
p->[1]->[2] q->[3]->[4]	p->[2]->[4] q->[1]->[3]	
p->[1] q->[2]->[3]->[4]->[5]	p->[2]->[1]->[4] q->[5]->[3]	

5. Array Programming, 10 points. Write a method called removeMax that removes the largest value from a list of integers. For example, if a variable called list stores this sequence of values:

```
[3, 1, 5, 7, 3, 19, 42, 8, 23, 7, 42, 2, -8, 9, 105, -3]
                                                  |
                                                 max
```

and the following call is made:

```
list.removeMax();
```

Then the largest value (105) is removed, leaving this list:

```
[3, 1, 5, 7, 3, 19, 42, 8, 23, 7, 42, 2, -8, 9, -3]
                        |
                       max
```

If the maximum occurs more than once, the method should remove the first occurrence. For example, if the same call is made again, the list becomes:

```
[3, 1, 5, 7, 3, 19, 8, 23, 7, 42, 2, -8, 9, -3]
```

You are writing a method for the ArrayIntList class discussed in lecture:

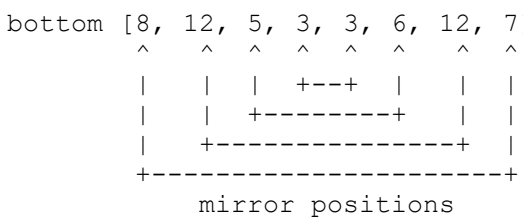
```
public class ArrayIntList {
    private int[] elementData; // list of integers
    private int size;          // current # of elements in the list

    <methods>
}
```

The method should throw an IllegalStateException if the list is empty because then there would be no maximum value to remove.

You may not call any other methods of the ArrayIntList class to solve this problem, you are not allowed to define any auxiliary data structures (no array, String, ArrayList, etc), and your solution must run in O(n) time where n is the length of the original list.

6. Stacks/Queues, 25 points. Write a method called makePalindrome that takes a stack of integers as a parameter and that removes pairs of values in mirror positions that don't match, resulting in a sequence of values that is a palindrome. A palindrome is a sequence of values that is the same in backwards order as it is in forwards order. For example, suppose that a stack s stores the following values:



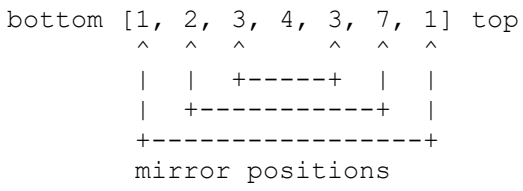
Suppose that we make the following call:

```
makePalindrome(s);
```

Notice that the first and last value are considered to be in mirror positions and they don't match (8 versus 7), so that pair will be removed. The second and second-to-last values are in mirror positions and they match (both 12), so they will not be removed. The third and third-to-last are in mirror positions and they don't match (5 versus 6), so that pair will be removed. The innermost pair matches (both 3), so it won't be removed. Thus, the stack ends up storing the two pairs that match:

```
bottom [12, 3, 3, 12] top
```

If the stack has an odd length, then the middle value should be retained because it can be part of a palindrome. For example, if the stack stored these values initially:



Then after the call, the stack should store the following values:

```
bottom [1, 3, 4, 3, 1] top
```

Notice that the middle value of 4 has been retained along with the two mirror pairs that match.

You are to use one queue as auxiliary storage to solve this problem. You may not use any other auxiliary data structures to solve this problem, although you can have as many simple variables as you like. You also may not solve the problem recursively. Your solution must run in O(n) time where n is the size of the stack. Use the Stack and Queue structures described in the cheat sheet and obey the restrictions described there (recall that you can't use the peek method or a foreach loop or iterator).