Practice Problems for CSC 20 Final Exam

(Spring 2018 – 5/4/18)

1. Array Programming:

Suppose that the integer array list has been declared and initialized as follows:

```
private int[] list = { 10, 20, 30, 40, 50 };
```

This statement sets up an array of five elements with the initial values shown below: **list**

Given this array, what is the effect of calling the method mystery(list);

if mystery is defined as:

```
public void mystery(int[] array) {
   int tmp = array[array.length - 1];
   for (int i = 1; i < array.length; i++) {
      array[i] = array[i - 1];
   }
   array[0] = tmp;
}
```

Work through the method carefully and indicate your answer by filling in the boxes below to show the final contents of list:

list

2. **Stacks/Queues:**

Write a method splitStack that takes a stack of integers as a parameter and splits it into negatives and non-negatives. The numbers in the stack should be rearranged so that all the negatives appear on the bottom of the stack and all the non-negatives appear on the top. In other words, if after this method is called you were to pop numbers off the stack, you would first get all the nonnegative numbers and then get all the negative numbers. It does not matter what order the numbers appear in as long as all the negatives appear lower in the stack than all the non-negatives. You may use a single queue as auxiliary storage. For example, if the stack stores [3, -5, 1, 2, -4] , an acceptable result would be [-5, -4 , 3, 1, 2]

3. Recursion Mystery. Consider the following method:

```
public int mystery3(int n)
{
  if (n < 0)
     return -mystery3(-n);
  else if (n < 10)
      return n;
  else
      return mystery3(n/10 + n % 10);
}
```

For each call below, indicate what value is returned:

Method Call	Value Returned
mystery3(6)	_____
mystery3(17)	_____
mystery3(259)	_____
mystery3(977)	_____
mystery3(-479)	_____

4. Recursion & Big O

Write a method called printStar(int n) which will print the following when n=4:

```
****
***
**
*
*
**
***
****
```

What is the Big O for this function?

5. Another Mystery Recursion:

```
public String mystery4(String s, char ch) {
   if (s.length() < 2) {
       return s;
   } else {
       char first = s.charAt(0);
       if (first == ch && first == s.charAt(1)) {
           return mystery4(s.substring(1), ch);
       } else {
           return s.charAt(0) + mystery4(s.substring(1), ch);
       }
   }
}
```

Trace this mystery4 method for the following: mystery4(“varriable”,”r”);

6. What is the asymptotic complexity of the following methods or code segments, in terms of the Big-O notation?

a)
```
void methodA(int n)
{
   for (int i=n; i>1; i-=2) {
      System.out.println(i);
   }
}
```

b)
```
void methodB(int n)
{
    for (int i=n; i<=n; i++) {
          for (int j=n; j>1; j=j/2) {
              System.out.println(j);
     }
}
```

c) Code segment:

```
int sum = 0;
int X = 100000;
for (int i = 1; i <= 4 * N; i++) {
  for (int j = 1; j <= X + 2; j++) {
    sum++;
  }
  for (int j = 1; j <= X * 100; j += 2) {
    for (int k = 1; k <= X * X; k++) {
      sum++;
    }
  }

  sum++;
}
System.out.println(sum);
```

d) Code segment:

```
int sum = 0;
for (int j = 0; j < 100 * N; j++) {
  for (int i = N; i > 0; i /= 2) {
    sum++;
  }
}
System.out.println(sum);
```

7. Practice in Inheritance and Polymorphism: Be sure to understand how this program works. Note there is no computer allowed on the exam. Please work this manually.

Assuming that the following classes have been defined:

```
public class One {
    public void method1() {
        System.out.println("One1");
    }
}

public class Two extends One {
    public void method3() {
        System.out.println("Two3");
    }
}

public class Three extends One {
    public void method2() {
        System.out.println("Three2");
        method1();
    }
}

public class Four extends Three {
    public void method1() {
        System.out.println("Four1");
        super.method1();
    }

    public void method3() {
        System.out.println("Four3");
    }
}
```

And assuming the following variables have been defined:

```
One var1 = new Two();
One var2 = new Three();
One var3 = new Four();
Three var4 = new Four();
Object var5 = new Three();
Object var6 = new One();
```

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();`	_____
`var2.method1();`	_____
`var3.method1();`	_____
`var4.method1();`	_____
`var5.method1();`	_____
`var6.method1();`	_____
`var4.method2();`	_____
`var4.method3();`	_____
`((Two)var1).method2();`	_____
`((Three)var1).method2();`	_____
`((Two)var1).method3();`	_____
`((Four)var2).method1();`	_____
`((Four)var3).method1();`	_____
`((Four)var4).method3();`	_____
`((One)var5).method1();`	_____
`((Four)var5).method2();`	_____
`((Three)var5).method2();`	_____
`((One)var6).method1();`	_____
`((One)var6).method2();`	_____
`((Two)var6).method3();`	_____

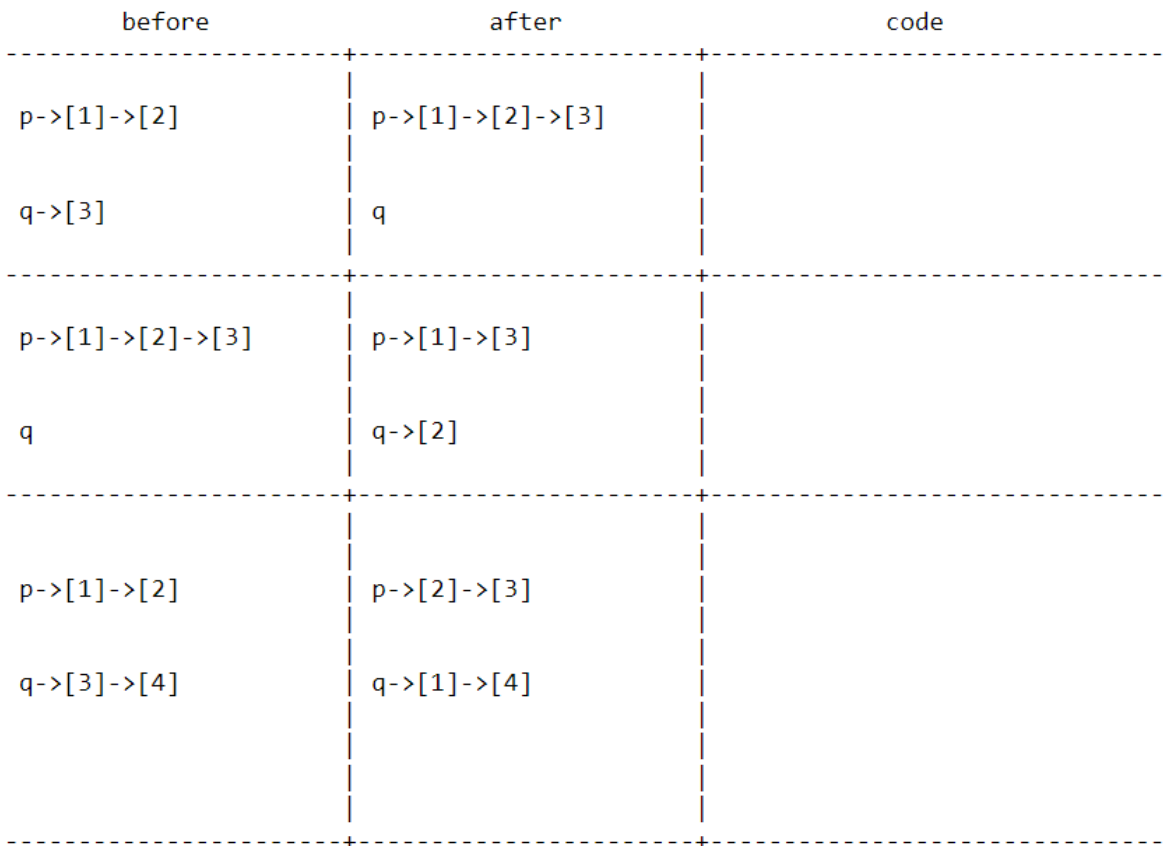
8. Linked Lists

```
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 as shown below:
```

```
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 LinkedList lab, 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]->[2]->[3]
q
p->[1]->[3]
q->[2]
p->[1]->[2]
q->[3]->[4]
p->[2]->[3]
q->[1]->[4]