Quiz 15: Recursion and object references SOLUTION

CSCI 110 Section 1

Friday, September 30, 2016

Say you have the following function:

```
void someFunction(double d) {
    if (d < 3) {
        System.out.println("pow");
        return;
    }
    someFunction(d - 1.5);
    System.out.println(d);
}
```

1) What’s printed when you call someFunction(6)? [no points, warmup]

```
pow
3
4.5
6
```

First, we call someFunction(6)

- d == 6 in here
- It's not true that d < 3 so we don't go with the base case
- The next thing we do is call someFunction(4.5)
    - d == 4.5 in here
    - It's not true that d < 3 so we don't go with the base case
    - The next thing we do is call someFunction(3)
        - d == 3 in here
        - It's not true that d < 3 so we don't go with the base case
        - The next thing we do is call someFunction(1.5)
            - d == 1.5 in here
            - d < 3 is true so we take the base case
            - **Print "pow"**
        - **Then we print d, which is 3**
    - **Then we print d, which is 4.5**
- **Then we print d, which is 6**

2) Write a recursive function that prints a countdown from a number, going down by two. Once the count is zero or below, it should print “BLASTOFF!”. [30 points]

What should the parameters be?
one integer will work so that we can do recursion

What should the return type be?
void - don't need to return anything; only print

```
void blastoff(int n) {
    if (n <= 0) {
         System.out.println("BLASTOFF!");
         return;
    }
    System.out.println(n);
    blastoff(n-2);
}
```

Consider this function:

```
void other(int m) {
    if (m <= 0) {
        System.out.println("ok");
        return;
    }
    System.out.println("wait");
    other(m - 1);
    other(m - 3);
    System.out.println(m);
}
```

3) What's printed when you call other(3)? [30 points]

```
wait
wait
wait
ok
ok
1
ok
2
ok
3
```

Remember that the CPU does one thing at once. We must do everything on one line before moving on to the next line.

Here's everything that happens. Printing bolded so you can follow along:

First, we call other(3)

- m == 3 in here
- Not true that m <= 0 so we don't take the base case
- **Print "wait"**
- Call other(2)
    - m == 2 in here

    - Not true that m <= 0 so we don't take the base case
    - **Print "wait"**
    - Call other(1)
        - m == 1 in here
        - Not true that m <= 0 so we don't take the base case
        - **Print "wait"**
        - Call other(0)
            - m == 0 in here
            - True that m <= 0 so take the base case
            - **Print "ok"**
            - Return
        - Call other(-2)
            - m == -2 in here
            - True that m <= 0 so take the base case
            - **Print "ok"**
            - Return
        - **Print m, which is 1**
    - Call other(-1)
        - m == -1 in here
        - True that m <= 0 so take the base case
        - **Print "ok"**
        - Return
    - **Print m, which is 2**
- Call other(0)
    - m == 0 in here
    - True that m <= 0 so take the base case
    - **Print "ok"**
    - Return
- **Print m, which is 3**

Consider these two functions that each take an ArrayList<Integer> as a parameter:

```
void something(ArrayList<Integer> ints) {
    ints.add(777);
}

void somethingElse(ArrayList<Integer> ints) {
    ints = new ArrayList<Integer>();
    ints.add(333);
}
```

4) What gets printed when the following code is run? [20 points]

```
ArrayList<Integer> ints = new ArrayList<Integer>();
ints.add(10);
something(ints);
System.out.println(ints);
```

[10, 777]

The call to something() modifies ints by adding one element, 777.

5) What gets printed when the following code is run? [20 points]

```
ArrayList<Integer> ints = new ArrayList<Integer>();
ints.add(1);
ints.add(2);
somethingElse(ints);
something(ints);
System.out.println(ints);
```

[1, 2, 777]

The call to somethingElse() does not modify its parameter. It creates and assigns a new, different ArrayList<Integer> to its parameter `ints`. Then it adds 333 to that new, different ArrayList<Integer>.

Then we call something() which modifies its parameter by adding 777.