Ch 8 review

Wednesday, April 17, 2013
9:38 AM

The next several questions refer to the following recursive method:

```
public static int compute(int x, int y) {
     if (x != y)
          return compute(x+1, y-1);
     else
          return x;
}
```

c(1,5): ~~c(2,4)~~ $\boxed{3}$

c(2,4): ~~c(3,3)~~ 3

c(3,3): 3

1. What is returned by the call `compute(1, 5)`?
   a. 1
   b. 2
   [✓ c. 3]
   d. 4
   e. No value is returned because infinite recursion occurs.

2. How many times would `compute` be called in the previous problem (including the initial call)? (free response)

   $\boxed{3}$

3. What is the condition for the base case in the recursive method?
   a. When `x` is not equal to `y`
   [✓ b. When `x` is equal to `y`]
   c. When `x` is less than `y`
   d. When `x` is greater than `y`
   e. When `x` is equal to 0.

4. Which of the following calls leads to infinite recursion?

   ~~I.~~ `compute(2, 8)` → 2,8 3,7 4,6 5,5 x
   [✓ II.] `compute(8, 2)` → 8,2 9,1 10,0 11,-1 ...
   [✓ III.] `compute(2, 5)` → 2,5 3,4 4,3 5,2 ...

   a. I only
   b. II only
   c. III only
   d. I and II
   [✓ e. II and III]

The next two questions involve the following recursive method. The method is designed to print every other positive number that is less than or equal to a given positive number; printing starts with either 1 or 2.

Base Case
n==0

```
public static void recur(int n) {
      if (n != 0) {
            recur(n - 2);
            System.out.print(n + " " );
      }
}
```

5. The `recur` method is prone to infinite recursion. What initial values of `n` would lead to infinite recursion? (free response)

Negative numbers
Odd numbers

6. What change could be made to the `if` statement to make the method resistant to infinite recursion? (free response)

if(n>0)

For the following questions consider the following recursive method, and assume that int[ ] a = {15, 10, 11, 16, 20, 18}

length = 6

```
public static int mysteryMethod(int[] a, int j) {
      if (j < a.length) {
            if (a[j] % 10 != 0)
                  return mysteryMethod(a, j+1);
            else
                  return mysteryMethod(a, j+1) + 1;
      }
      else
            return 0;
}
```

$\boxed{2}$

mm(a,0): ~~mm(a,1)~~

1

mm(a,1): ~~mm(a,2)~~ + 1 → 2

1

mm(a,2): ~~mm(a,3)~~

1

mm(a,3): ~~mm(a,4)~~

0

mm(a,4): ~~mm(a,5)~~ + 1 → 1

mm(a,5): ~~mm(a,6)~~ 0

mn(a,6): 0

```
public static int someMethod(int[] a, int j) {
      if (j < a.length) {
            return a[j] + someMethod(a, j+2);
      }
      else
            return 0;
}
```

15/10/11/16/20/18
↑ ↑ ↑

31

sm(a,0): 15 + ~~sm(a,2)~~ → $\boxed{46}$

sm(a,2): 11 + ~~sm(a,4)~~ 20 → 31

sm(a,4): 20 + ~~sm(a,6)~~ 0 → 20

sm(a,6): 0

7. What would a call to `mysteryMethod(a, 0)` result in?
(Free Response)

$\boxed{2}$ (Counts the multiples of 10 starting at index j)

8. What would a call to `mysteryMethod(a, 2)` result in?
(Free Response)

$\boxed{1}$

9. What would a call to `someMethod(a, 0)` result in?
(Free Response)

$\boxed{46}$ (it adds every other number starting at index j)

10. What would a call to `someMethod(a, 1)` result in?
(Free Response)

$10 + 16 + 18 = \boxed{44}$

11. What would a call to `someMethod(a, 2)` result in?
(Free Response)

$11 + 20 = \boxed{31}$