Retake 2 Quiz Answers

Question 1. Consider the following method. What integer is returned by the call `foo("a1B2c3D")`?

```
public int foo(String s) {
    if (s.length() == 0) return 0;
    char c = s.charAt(0);
    int add = Character.isDigit(c) ? 1 : 0;
    return add + foo(s.substring(1));
}
```

Question 2. Consider the following method. What string is returned by the call `bar("WXYZAB")`?

```
public String bar(String s) {
    if (s.length() <= 2) return s;
    return s.charAt(s.length() - 1) + "" + bar(s.substring(1, s.length() -
1)) + s.charAt(0);
}
```

Question 3. Consider the following method. What value is returned by the call `g(nums, 3)`?

```
public static int g(int[] a, int n) {
    if (n == 1) return a[0];
    else return a[n-1] + g(a, n-1);
}
```

Assume `int[] nums = {4, 1, 6, 2};`.

Question 4. Examine the following recursive method. What integer is returned by the call `mystery2(arr, 4)`?

```
public static int mystery2(int[] a, int n) {
    if (n == 1) return a[0] * 2;
    a[n-1] = a[n-1] - 1;
    return a[n-1] * mystery2(a, n-1);
}
```

Assume `int[] arr = {5, 6, 7, 8};` before the call.

Question 5. Consider the following method. After executing `invert(arr, 0)`, what is the content of `arr`?

```
public static void invert(int[] a, int left) {
    int right = a.length - 1 - left;
    if (left >= right) return;
    int temp = a[left];
    a[left] = a[right];
```

```
    a[right] = temp;
    invert(a, left + 1);
}
```

arr is initially {10, 20, 30, 40, 50, 60}.

Question 6. Consider the following method. After executing f(a, 5), what is the value stored in a[5]?

```
public static void f(int[] a, int n) {
    if (n == 0) {
        a[0] = 1;
        return;
    }
    f(a, n - 1);
    a[n] = n * a[n - 1];
}
```

a is an int array of length at least 6, initially all zeros.

Solutions and Rubrics

Question 1

Step-by-step evaluation of countDigits("a1B2c3D"): 1. First character 'a' → not a digit → add = 0 → recurse on "1B2c3D" 2. '1' → digit → add = 1 → recurse on "B2c3D" 3. 'B' → not digit → add = 0 → recurse on "2c3D" 4. '2' → digit → add = 1 → recurse on "c3D" 5. 'c' → not digit → add = 0 → recurse on "3D" 6. '3' → digit → add = 1 → recurse on "D" 7. 'D' → not digit → add = 0 → recurse on "" (empty string) 8. Base case returns 0. Summing the adds: 0 + 1 + 0 + 1 + 0 + 1 + 0 + 0 = **3**. Therefore countDigits("a1B2c3D") returns 3.

Question 2

Step-by-step trace: 1. Call bar("WXYZAB") – length 6 > 2. - last char = 'B', first char = 'W'. - Recursive call on s.substring(1,5) → "XYZA". 2. Call bar("XYZA") – length 4 > 2. - last char = 'A', first char = 'X'. - Recursive call on s.substring(1,3) → "YZ". 3. Call bar("YZ") – length 2 ≤ 2, base case returns "YZ". 4. Unwind: - bar("XYZA") returns 'A' + "YZ" + 'X' → "AYZX". - bar("WXYZAB") returns 'B' + "AYZX" + 'W' → **"BAYZXW"**. Thus the method call bar("WXYZAB") evaluates to "BAYZXW".

Question 3

Trace the recursion step-by-step: - g(nums,3) → a[2] + g(nums,2) → 6 + g(nums,2) - g(nums,2) → a[1] + g(nums,1) → 1 + g(nums,1) - g(nums,1) → base case returns a[0] → 4 Now unwind: - g(nums,2) = 1 + 4 = 5 - g(nums,3) = 6 + 5 = 11 Therefore the method returns **11**.

Question 4

Trace the recursion step-by-step: 1. `mystery2(arr,4)`: decrement `a[3]` → 8-1 = 7; return `7 * mystery2(arr,3)`. 2. `mystery2(arr,3)`: decrement `a[2]` → 7-1 = 6; return `6 * mystery2(arr,2)`. 3. `mystery2(arr,2)`: decrement `a[1]` → 6-1 = 5; return `5 * mystery2(arr,1)`. 4. Base case `mystery2(arr,1)`: return `a[0] * 2` → 5 * 2 = 10. Now unwind: - From step 3: `5 * 10 = 50`. - From step 2: `6 * 50 = 300`. - From step 1: `7 * 300 = 2100`. Therefore the method returns **2100**.

Question 5

The routine swaps the element at the current left index with the element at the symmetric right index and then recurses inward.

Step-by-step: 1. left=0, right=5 → swap 10↔60 → `{60, 20, 30, 40, 50, 10}` 2. left=1, right=4 → swap 20↔50 → `{60, 50, 30, 40, 20, 10}` 3. left=2, right=3 → swap 30↔40 → `{60, 50, 40, 30, 20, 10}` 4. left=3, right=2 → base case (`left >= right`) stops. The final array is `[60, 50, 40, 30, 20, 10]`.

Question 6

The method fills the array with factorial values recursively.

1. Call `fillFact(a,5)`. Since `n != 0`, it recurses with `n = 4`.
2. This pattern continues until the base case `n == 0` is reached, where `a[0]` is set to 1.
3. Returning from each recursive call, the assignment `a[n] = n * a[n-1]` is executed:
   - n = 1 → a[1] = 1 * a[0] = 1 * 1 = 1
   - n = 2 → a[2] = 2 * a[1] = 2 * 1 = 2
   - n = 3 → a[3] = 3 * a[2] = 3 * 2 = 6
   - n = 4 → a[4] = 4 * a[3] = 4 * 6 = 24
   - n = 5 → a[5] = 5 * a[4] = 5 * 24 = 120
4. Thus after the call finishes, `a[5]` holds `120`.

The recursion works because each call relies on the already-computed factorial for the previous index.