**HW4**

**To Turn in: please submit the questions and your answers below them in a pdf file on canvas.**

**Perform a time-complexity (Big-O) analysis for each of the next three problems (problems 1, 2, and 3). For full credit you should be able to produce a logical justification for your answer (a growth rate function can help demonstrate this – but is NOT required – so at least show in general why the Big-O is what it is). Equations you may need: (1) 1 + 2 + 3 + 4 + …+ n = (1 + n) \* n / 2; (2) 1 + a + a2 + a3 + … + an = (an+1 – 1) / (a-1).**

**1. (40 Points)** **public static void** two(int n)  
 {  
 **if**(n > 0)  
 {  
 System.out.println("n: " +n);  
 two(n - 1);  
 two(n - 1);  
 }  
 **else** **if** (n < 0)  
 {  
 two(n + 1);  
 two(n + 1);  
 System.out.println(″n: ″ + n);  
 }  
 }

If n is positive, we enter this loop  
  
 if(n > 0)  
 {  
 System.out.println("n: " +n);   
 two(n - 1);  
 two(n - 1);  
 }  
  
Recursive Tree  
 Calls  
 two(n) (1)  
 / \  
 two(n–1) two(n–1) (2)  
 / \ / \  
 two(n-2) two(n-2) two(n-2) two(n-2) (4)  
 | | | |  
… two(n-n) two(n-n)… two(n-n) two(n-n) (2^n)  
  
The bottom level is dominant in the GRF, so big\_oh = O(2^n).

**2. (30 Points)**  
**public void** three(**int** n)

{  
 **int** i, j, k;

**for** (i = n/2; i > 0; i = i/2)

**for** (j = 0; j < n; j++)

**for** (k = 0; k < n; k++)

System.out.println("i: " + i + " j: " + j+" k: " + k);

} // end three

In the for loop we have i = i/2, so i will be cut in half for every iteration. The GRF will be similar to that of log(n) which is the dominant factor for all of the 3 loops. Big\_oh = O(log\_2(n)).

**3. ( 30 points)**

**public** **static** **void** four(**int** n)

{

**if** (n > 1) (n-1)

{

System.out.println(n); (n-1)

four(n-1); (n-1)  
 }

**for** (**int** i = 0; i < n; i++) (1)+(n+1)+(n)

System.out.println(i); (n)

}

As we enter the if statement, there is a recursive call so it will occur n-1 times since n > 1. Same for each statement in the if.

There will be two parts to the GRF since there exists a for loop after the if statement.  
  
GRF for the if is: n – 1 + n – 1 + n – 1 = 3n – 3

GRF for the for is: 1 + n + 1 + n + n = 3n + 2

We then drop the constants from both equations and we end up with n for both, then we multiply them together, thus getting

GRF = n^2

So big\_oh = O(n^2)