**Problem 1.**

Asymptotic running time: O(n) + O(n2) = O(n2).

**Problem 2.**

a.

***Algorithm***: Merge(array1, array2)

***Input***: array1, array2 sorted array of elements

***Output***: newArray merge to sorted array

end1 ← array1.length - 1

end2 ← array2.length – 1

newLength ← array1.length + array2.length

while (newLength > 0 ) do

newLength ← newLength – 1

if(end2 < 0 || (end1 >= 0 && array1[end1] > array2[end2]))

newArray[newLength] ← array1[end1--]

else

newArray[newLength] ← array2[end2--]

return newArray

b. Asymptotic running time: O(n)

c. See file Problem2.java

**Problem 3:**

a) c = 5 and n0 = 1. Whenever n ≥ n0, 1 + 4n2 ≤ n2 + 4n2 = 5n2 = cn2

b) Given postive c and natural number n0, we find n ≥ n0 so that n2 −2n6 ≤ cn:

Let n be such that n > max{c + 2, n0}. Then n2 − 2n > cn if and only if n − 2 > c if and only if n > c + 2.

That last inequality is true, so the first one is also true.

c) We can say that since the limits at ∞ of log(n) and n “exist” (limits are ∞ in each case), we may use the limit version of the defintion of o. Doing so, we get

limn→∞ log n/n = limn→∞ (c/n)/1 = limn→∞ c/n = 0.

Therefore, log n is o(n)

d) Find c > 0 such that for every positive integer n0 there is a positive integer n ≥ n0 such that n > cn. We choose c = 1/2. Then for any n (in particular, for any n ≥ n0 for any choice of n0), we have n > cn, as required.