2) (10 pts) ANL (Sorting)

Jesse has mistakenly written his Merge Sort so that instead of making recursive calls on each half of the array (code below), he calls a function that runs an Insertion Sort on each half of the array. You may assume the function insertionSort runs properly and executes the steps of an Insertion Sort. He tests his function on an array of size 100,000 in reverse sorted order, and discovers that instead of running in under one second, his code takes 9 seconds. How long (in seconds) would sorting the same array (100,000 elements in reverse order), ***on the same computer***, using a single Insertion Sort, be expected to take?

To earn full credit, you must justify your answer by looking at the number of simple operations in both algorithms and comparing the differences in multiplicative constants between the two algorithms.

```
void mergeSort(int array[], int low, int high) {
    if (low >= high) return;
    int mid = (low+high)/2;
    insertionSort(array, low, mid);
    insertionSort(array, mid+1, high);
    merge(array, low, mid, high);
}
```

Let the T(n) be the run time of insertion sort on an array of size n. We know that for some constant c, $T(n) = cn^2$. The cost of a single merge on two arrays of size n/2 is O(n). Let S(n) = dn, for some constant d be the run time of a Merge. This means the run time of the code for "mergeSort" written above would take this much time to sort an array of size n.

$$\boldsymbol{2T\left(\frac{n}{2}\right) + S(n)}$$

For our situation we have n = 100,000 and the amount of time the code took is 9 seconds, plugging in, we find:

$$2c(50000)^2 + d(100000) = 9\ sec$$
$$2(2500000000)c + d(100000) = 9\ sec$$

Clearly, there are an infinite number of solutions for c and d to this equation, since there is only one equation. But, it's fairly reasonable to assume two things: (1) the constants are similar, and (2) the multiplier of c is so much smaller than the multiplier for d, that the second term adds a negligible amount of time (one ten-thousandth roughly, so if I am answering in seconds, there is no change), so for the purposes of this estimation, we can remove that term, so we have: $\boldsymbol{5,000,000,000c = 9\ sec.}$

If we were to do one Insertion sort on an array of size 100000, our run time would be $c(100{,}000)^2$, so we have:

$$T(100{,}000) = \frac{9sec}{5 \times 10^9} \times (10^5)^2 = \boldsymbol{18\ seconds}$$

Grading: 5 pts for recognizing that the bulk of the code is 2 insertion sorts on arrays of size n/2. 5 pts for comparing the run time of that to 1 insertion sort of an array of size n and recognizing that the latter takes roughly double the time.