John Pennig

Hwork3

Question 2:

1. If there are 0 items left in the non-processed list, the function returns. If there is 1 item left in the non-processed list, the function pushes that item to the left partition and returns. These are the base cases to stop recursion.
2. Recursively partition the half's until they only contain one element. Once they contain one element, they are technically sorted. Then merge all the separate pieces back together while maintaining order.
3. The end conditions are that one of the lists is empty. If so, it returns the other list since its already sorted. If both lists are not empty, then the function compares the heads of the lists to see which one is smaller. Append the smaller item to the processed list and recurse to continue until lists are empty.

Question 4:

a. When we start insertion sort, we have an unsorted list and a sorted list. The unsorted list should begin full with every element in it, and the sorted list should be completely empty. We can use this information to decide termination when the unsorted list is completely empty.
b. The process is trivially accomplished when the unsorted list is empty. The recursive call will move one item from the unsorted list into the sorted list and then call the function again with the updated lists.

c. The two ways that the process terminates is if the insertion spot has been found (value is smaller than the item after), or if there is no larger item and the spot is at the end of the list.