1) (10 pts) ANL (Algorithm Analysis)

Consider the following function that takes a list of **unique** integers for input

1. The list is empty return 0
2. Choose a random integer, `x`, from the list
3. Place every value from the list less than the value `x` in list 1 every value greater than `x` in list 2.
4. Run this function on list 1 and store the result in a variable called `left_answer`.
5. Run this function on list 2 and store the result a variable called `right_answer`.
6. Add to the answer the function run on list 2
7. Return the value `left_answer + right_answer + x`

What is the best case runtime and the worst case runtime for the above function, in terms of the input list size, **n**? What standard algorithm taught in COP 3502 is this algorithm closest to? In terms of that standard algorithm, what does the return value of this function represent? Provide proof of the runtimes, but in doing so, you may use results about known algorithms from COP 3502 without proving those results.

The above method recursively partitions the list into two pieces after removing one element. This behavior is similar to how quick sort works. The random number selected is like the pivot/partition element. When the pivot is close to the middle the algorithm has the best performance which is O(nlogn) where n is the size of the array, because the recurrence relation governing the run-time is $T(n) = 2T(n/2) + O(n)$, which is the same recurrence relation for Merge Sort, which has a solution of O(nlgn). The worst case behavior is if the pivot/partition element chosen each time is either the smallest or largest in the list. This gives a recurrence relation of $T(n) = T(n-1) + O(n)$ for the worst case behavior, which leads to a worst case run time of $O(n^2)$. The return value of the function is simply the sum of all the values in the array.

Grading: 2 pts for best case result w/o any proof
2 pts for worst case result w/o any proof
2 pts for identifying quick sort as the similar algorithm
2 pts for stating that the return value is the sum of all the values in the input list
2 pts for the justification of the run-times, which can simply be stating the best and worst case run times of quick sort, since the structure is the same.