Name:_____

1. *Quick sort* is another advanced sort that often is quicker than merge sort (hence its name). The general idea is as follows. Assume "n" items to sort.

- Select a "random" item in the unsorted part as the *pivot*
- Rearrange (called *partitioning*) the unsorted items such that:

All items <= to Pivot	Pivot Item	All items > to Pivot

- Quick sort the unsorted part to the left of the pivot
- Quick sort the unsorted part to the right of the pivot

Before Spring Break, we developed the following partial code for the quickSort and partition:

```
# Recursive quick sort to sort in ascending order
def quickSort(myList, start, end):
    if start < end:
        pivotIndex = partition(myList, start, end)
        quickSort(myList, start, pivotIndex-1)
        quickSort(myList, pivotIndex+1, end)


def partition(myList, start, end):
    pivot = myList[start]
    left = pivot + 1
    right = end
    while left <= right:

        # continue to move left looking for an item bigger than pivot
        while left <= right and myList[left] <= pivot:
            left = left + 1

        # continue to move right looking for an item <= pivot
        while left <= right and myList[right] > pivot:
            right = right - 1
```

a) The second inner while loop's (the one for moving right) first condition is unnecessary, why?

b) Complete the partition code above.

Name:_____

2. Now, let's consider calling quickSort with the following list:

```
def main( ):
    aList = [ 60, 35, 10, 40, 45, 20, 25, 50 ]
    quickSort(aList, 0, len(aList)-1)
```

To trace quickSort code we'll draw a "call tree" like we did for mergeSort.