These practice problems are all on recursion.

1. What is the output produced by the following function when it is called as
   \( \text{partition}([6, 1, 9, 3, 2, 6, 1, 4], 0, 7) \)

   ```python
def partition(L, first, last):
    p = first

    for current in range(p+1, last+1):
        print L[first:p], L[p], L[p+1:current]
        if L[current] < L[p]:
            swap(L, current, p+1)
            swap(L, p, p+1)
        p = p + 1

    return p
```

2. This question is based on understanding the working of the following implementation of
   the quick sort algorithm.

   ```python
def generalQuickSort(L, first, last):
    if first < last:
        p = partition(L, first, last)
        generalQuickSort(L, first, p-1)
        generalQuickSort(L, p+1, last)

    # Insert the code
    if first > last:
        print "Base Case 0"

    # Right at the beginning of the function. How many times will we see "Base Case 0"
    # printed if we make the call
    generalQuickSort([6, 1, 9, 3, 2, 6, 1, 4], 0, 7)

(b) Insert the statement
    ```python
    print L[first:p], L[p], L[p+1:last+1]
    ```
    just after the line of code \( p = \text{partition}(L, \text{first}, \text{last}) \) in the above function. What output do we get if make the call
    \( \text{generalQuickSort}([6, 1, 9, 3, 2, 6, 1, 4], 0, 7) \)

(c) How many calls in total are made to the function \( \text{generalQuickSort} \) if it is called as
    \( \text{generalQuickSort}([6, 1, 9, 3, 2, 6, 1, 4], 0, 7) \)

(d) Delete the second (recursive) call to \( \text{generalQuickSort} \) in the above definition of
    \( \text{generalQuickSort} \). Start with a list \( L = [6, 1, 9, 3, 2, 6, 1, 4] \) and call this
    function as
    \( \text{generalQuickSort}(L, 0, 7) \)

    What is \( L \) after this call?
3. How many times is the \texttt{swap} function called (from \texttt{partition}) as a result of the call 
\texttt{generalQuickSort([6, 5, 4, 3, 2, 1], 0, 5)}

4. How many times is the \texttt{swap} function called (from \texttt{partition}) as a result of the call 
\texttt{generalQuickSort([1, 2, 3, 4, 5, 6, 7, 8], 0, 7)}

5. Write down a length-7 list sequence that causes \texttt{partition} to split the list into exactly two halves each time \texttt{partition} is called as part of the call to \texttt{generalQuickSort} on this list. In other words, the first time \texttt{partition} is called, it is called on a length-7 list, and it should split the list into two sublists of size 3 each. Subsequently, \texttt{partition} will be called on two length-3 lists. In each case, \texttt{partition} should split the list into two length-1 lists.