Computer Science AP
Merge Sort

```
private void mergeSort( int[] a, int[] tmpArray, int left, int right ) {
        System.out.println("mergeSort:   Left is " + left + ", right is " + right);
        if( left < right ) {
            int center = ( left + right ) / 2;
            mergeSort( a, tmpArray, left, center );
            mergeSort( a, tmpArray, center + 1, right );
            merge( a, tmpArray, left, center + 1, right );
        }
}
```

Complete the line that would 'kick off' the sort:

```
int[] list = {5,1,90,80} ;
int[] temp = new int[list.length];
mergeSort(     ,      ,      ,      ) ;
```

Including the first call to mergeSort, how many times does mergeSort get called during the recursive process of sorting the list above?

What are the first two changes to values in list as the sort works?

What is the last value in list to change when the sort finally ends?

How many times will merge be called during the sorting process?

A programmer reverses lines in mergesort so that it reads as shown below.
Will the sort still work?
If it doesn't, why not.
If it does, how will the sorting process be different?

```
            mergeSort( a, tmpArray, center + 1, right );
            mergeSort( a, tmpArray, left, center );
```

How many times would the mergesort method be called (over the entire sort) for a list with length n where n=2, n=4, n=8, n=16

Do you see a pattern?

How do the numbers of calls to mergesort change as the size of the list is doubled?

Mergesort is often refered to as a method that 'divides and conquers'. Explain why this description is apropriate.