Sorting

1) Show the contents of the array below being sorted using Selection Sort at the end of each loop iteration:

Initial	3	9	4	5	6	1	3	8	7
Sorted									

2) Show the contents of the array below being sorted using Insertion Sort at the end of each loop iteration:

Initial	3	9	4	5	6	1	3	8	7
Sorted									

3) Show the contents of the array below being sorted using Bubble Sort at the end of each loop iteration:

Initial	3	9	4	5	6	1	3	8	7
Sorted									

4) Complete the following Merge function by filling in the blanks below.
It should merge the sorted values from start to middle, with the sorted values from middle to end.

start middle end

2	7	16	44	55	89	1	6	9	13	15	49

After:

start middle end

1	2	6	7	9	13	15	16	44	49	55	89

```
// Pre-condition: start,middle and end are valid indexes to the array,
//                with middle >= start and middle <= end.
//                 The values from index start to middle are sorted AND
//                 the values from middle+1 to end are sorted.
// Post-condition: The array from start to end is sorted.

void Merge(int values[], int start, int middle, int end) {
    int *temp, i, length, count1, count2, mc;

    // Allocate the proper amount of space for our temp array.
    length = _____________________________;
    temp = (int*)calloc(length, sizeof(int));


    count1 = _____________;  // These will be our indexes into our 2 sorted lists.
    count2 = _____________;
    mc = 0;                  // Keeps track of our index into our temp array.

    // Here we copy values into our temp array, if there are any to copy.
    while ((count1 < middle) || (count2 <= end)) {

        // Next value to copy comes from list 1 - make sure list
        // 1 isn't exhausted yet. Also make sure we don't access index
        // count2 if we aren't supposed to.
        if (count2 > _____________ ||
           (count1 < _____________ && ____________________________________)) {
            temp[mc] = values[count1];
            count1++;
            mc++;
        }
        // We copy the next value from list two.
        else {
            temp[mc] = _____________________________;
            _____________++;
            _____________++;
        }
    }

    // Copy back all of our values into the original array.
    for (i=start; i<=end; i++)
        values[i] = temp[i - start];

    // Don't need this space any more, so let's free it!
    free(_____________);
}
```