1. What is big-O complexity? Why is it a useful measure?

2. What is the big-O complexisty of the following procedure?
   Look at each element in a list of length $n$ and return the biggest

3. On the *ith* step of insertion sort, what can we say about the front of the list?

4. In one or two sentences, describe the difference between insertion and selection sort.

5. Consider the following method. What is returned by the call $test(15, 4)$?

```
def test(a, b):
    if (a < b):
        return 0
    else
        return 1 + test(a-b, b)
```

6. Trace the execution on insertion sort on the following list:

```
[20, 25, 15, 9, -8, 12, -10, 5]
```

7. Trace the execution on selection sort on the following list:

```
["grape", "apple", "orange", "banana", "seaweed", "milk"]
```

8. Trace the execution on merge sort on the following list:

```
[12, 0, -5, 16, -20, 57, 19, 3]
```

9. Assume the *merge* method is written:

```
def merge(left, right):
    left_point = 0
    right_point = 0
    merged = []
    while left_point < len(left) and right_point < len(right):
        if left[left_point] < right[right_point]:
            merged.append(left[left_point])
            left_point +=1
        else:
            merged.append(right[right_point])
            right_point +=1
    if left_point >= len(left):
        merged = merged + right[right_pointer:]
    else
        merged = merged + left[left_pointer:]
    return merged
```

Now, try to write the recursive method *mergeSort* that takes in a list and returns the list in sorted order. We did this in class, so try to do it without your notes first, then check to see how you did.