CS 62 Quiz

Mergesort

Mawhorter Spring 2017

Name: _____

1. Look at the implementation of `mergesort` below. It avoids the use of a temporary array to store merged values, instead using `ArrayList.remove` and `ArrayList.add` to move values around to where they need to be.

```
public void mergesort(ArrayList<Integer> values, int low, int high) {
    mergesort(values, low, (high - low) / 2)
    mergesort(values, (high - low)/2, high)
    merge(values, low, (high - low)/2, high)
}

// Doesn't require an extra storage array, which saves time
public void merge(ArrayList<Integer> values, int low, int mid, int high) {
    while (low < mid && mid < high) {
        if (values.get(low) <= values.get(mid)) {
            low += 1; // do nothing, low value remains in place
        } else { // move a value from the mid to before the low
            int tmp = values.remove(mid);
            values.add(low, tmp); // add balances remove
            low += 1;
            mid += 1; // because of the shift created by the add
        }
    }
    // Don't need to do anything here because remaining low or mid values
    // are already where they needed to be.
}
```

This version of `mergesort` turns out to be much slower than the original, especially for long lists. **Explain why**, and **give the big-O run time** for this version of `mergesort`. *(You may use the back of the page for your answer.)*