CS 112 mid term 2. Jae Hong Lee

Question 2.

```
public static String process (String s){
    if ( s.equal("")){
        return "";
    }else {
        String rest = process (S.subString (1));
        return S.charAt(0) + rest + S.charAt(0);
    }
}
```

Jae Hong Lee

Question 3

3.1

a) {10, 17, 51, 35, 30, 43, 46, 57}

b) {10, 17, 51, 35, 30, 43, 46, 57}

c) {10, 17, 30, 35, 51, 43, 46, 57}

3.2

a) {13, 35, 36, 15, 30, 17}

b) {46, 57, 63, 56}

c) {13, 15, 17, 30, 35, 36, 46, 57, 63, 56}

3.3

a) merge sort - Since the merge sort makes an additional temporary array of the same size as the original one so it takes the most additional space

b) Wrong, Since insertion sort use one loop from first index to the last the best case of insertion sort has a runtime of O(n), however Quick sort divide and conquer the best case of quicksort is O(n log n) so it is not true.

Question 4

Jae Hong Lee

4.1

$O(n^3)$ - the worst case is when arr has n length the run time of the most outer loop is $n/2$, the middle loop is n and the last loop is also n from the middle loop. So the total runtime is $n/2 \times n \times n = \frac{n^3}{2}$, so the runtime is $O(n^3)$

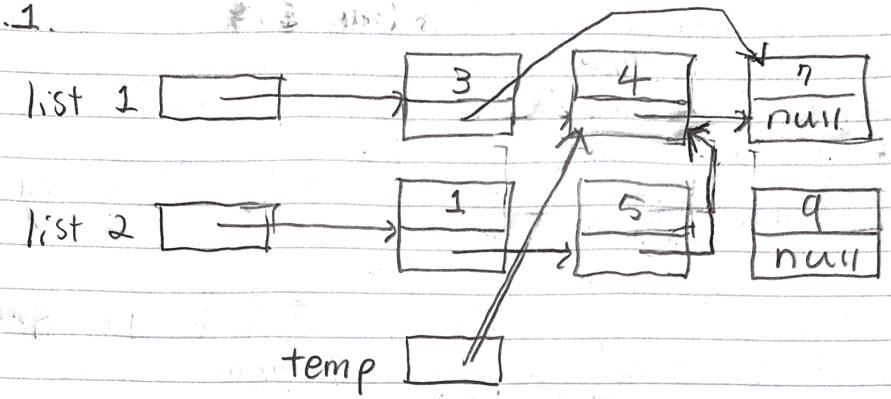
4.2

The best case time efficiency is $O(1)$, when the array length is 1. the worst case efficiecy of the algorith with 1 length array is 1. so the best case is $O(1)$

Question 5

Jae Hong Lee

Q5.1.



Q5.2

```
public static Node combine(Node list1, Node list2){
    if (list1 == null){
        return list2;
    } else if(list2 == null){
        return list1;
    }
    Node n = new Node();
    while (list1.next != null && list2.next != null){
        if (list1.val < list2.val){
            n.next = list1.next;
            list1 = list1.next;
        } if (list1.val > list2.val){
            n.next = list2.next;
            list2 = list1.next;
        }
    } return n;
}
```

Question 6

Jae Hong Lee

Q6.1

```
public static boolean doubled(StrNode s){
    StrNode trav = s;
    StrNode trail = null;

    if (s == null){
        return false;
    } else {
        while (trav != null){
            if (trav.ch == trail.ch){
                return true;
            }
            trail = trav;
            trav = trav.next;
        }
    } return false;
}
```

Q 6.2

Jue Hong Lee

```
Public Static int compare (StrNode S1, StrNode S2) {
    if (S1.ch < S2.ch) {
        return 2;
    } else if (S1.ch > S2.ch) {
        return 2;
    } else {
        compare (S1.next S2.next) {
    } return 0;
}
```

Question 7

Jae Hong Lee

Q7.1

O(1), from the LLlist when i is the first item, so x1 and x2 remove the first item and add the x1 * x2 to the front of the list then there is no need to walk down the Linked List

Q7.2

O(1), the best case is when i is the last index, so Removing the last item and adding to the last item does not require and shifting of array. so the best case scenario is O(1)

Q7.3

```
public void process(int i){
    int x1 = item[i];
     this.removeItem(i);
    this.addItem(this, i);
}
```