Problem 1: digitMatch

```
public static digitMatch(int x, int y) {
    // Throw an IAE if x < 0 or y < 0
    if (x < 0 || y < 0) {
        throw new IllegalArgumentException();
    }

    // BC: There's only one digit pair
    if (x < 10 || y < 10) {
        if (x % 10 == y % 10) {
            return 1;
        } else {
            return 0;
        }
    } else if (x % 10 == y % 10) { // RC: There's more than one digit pair
        return 1 + digitMatch(x / 10, y / 10);
    } else {
        return digitMatch(x / 10, y / 10);
    }
}
```

Problem 2: switchPairs

Solution 1:

Type your solution here:

```
public void switchPairs () {
    ListNode curr = this.front;
    ListNode prev = this.front;
    while (curr != null && curr.next != null) {
        // Before Swap
        ListNode temp = curr.next;

        // Swapping
        curr.next = temp.next;
        temp.next = curr;

            // Front Case
        if (curr == this.front){
            this.front = temp;
        }
        else {
            prev.next = temp;
        }

        // After Swap
        prev = curr;

        curr = curr.next;
    }
}
```

Solution 2:

```
public void switchPairs() {
    ListNode curr = front;

    if (front != null && front.next != null) {
        ListNode temp0 = front.next.next;

        front.next.next = front;
        front = front.next;
        front.next.next = temp0;

        curr = front.next;

        while (curr.next != null && curr.next.next != null) {
            ListNode temp = curr.next.next.next;
            curr.next.next.next = curr.next;
            curr.next = curr.next.next;
            curr.next.next.next = temp;
            curr = curr.next.next;
        }
    }
}
```

Problem 3: trim

Type your solution here:

```java
public void trim(int min, int max) {
    overallRoot = trimHelper(overallRoot, min, max);
}

private IntTreeNode trimHelper(IntTreeNode root, int min, int max) {
    if (root != null) {
        if (root.data < min) {
            root = trimHelper(root.right, min, max);
        } else if (root.data > max) {
            root = trimHelper(root.left, min, max);
        } else {
            root.left = trimHelper(root.left, min, max);
            root.right = trimHelper(root.right, min, max);
        }
    }
    return root;

}
```