Please complete the homework with your lab3 group. Total points is 17. The skeleton of the code can be founded at: http://gitlab.cs.ship.edu/chuo/swe300-in-class-java/-/tree/difference-list-hw6 You don't need vpn to access gitlab.

1. (2 points) Implement `printList` in Node.java. Note that `toString` is already given to you. Once finished, uncomment line 11-13 in Main.java. You should be able to see the expected results.

```
public static void printList(Node lst) {
  System.out.print(lst.toString());
  if (lst.next ≠ null) {
    printList(lst.next);
  } else {
    System.out.println();
  }
}
```

2. (2 points) Implement `equal` in Node.java. Once finished, uncomment line 19,22,23 in Main.java. You should be able to see the expected results.

```
public static boolean equal(Node l1, Node l2) {
  if (((l1 ≠ null) && (l2 ≠ null)) && (l1.val == l2.val)) {
    if ((l1.next ≠ null) && (l2.next ≠ null)) {
      return equal(l1.next, l2.next);
    } else {
      return true;
    }
  } else {
    return false;
  }
}
```

3. (3 points) Implement `append` in Node.java. (Hint: What if `l1` is `null`?).

```
public static Node append(Node lst1, Node lst2) {
  if (lst1 ≠ null) {
    return append(lst1.next, lst2);
  } else {
    lst1 = lst2;
    return lst1;
  }
}
```

4. (3 points) Given the following code:

```
1 Node l1 = makeList(0);
2 l1 = append(l1, makeList(0));
3 l1 = append(l1, makeList(0));
```

If I change line 2 and line 3 to:

```
l1 = append(makeList(0), l1);
l1 = append(makeList(0), l1);
```

Will the resulting list be different? Is there any difference in speed? Why?

The resulting lists would not be different. There is no difference in speed, as both orders report an elapsed time of 0ms. There is not difference in speed since both lists contain the maximum nodes N, so the runtime will be the same for both appends.

5. (3 points) Assume f, g, h are difference lists constructed from three linked lists, l1, l2, l3.

```
var f = toDiff(l1);
var g = toDiff(l2);
var h = toDiff(l3);
```

Let `i = DiffList.append(f,g) = x -> f.f.apply(g.f.apply(x))`. Demonstrate that the evaluation of `DiffList.append(i, h)` is actually right-associative.

Let i = DiffList.append(f,g) = x → f.f.apply(g.f.apply(x));
x → f.f.apply(g.f.apply(x))

this is F compose G, f(g(x))
so g.f.apply(x) must run before f.f.apply(..) can occur
therefore DiffList.append(i,h) is right associative

6. (4 points) Use `1024` for both `CHUNK_SIZE` and `N`. Run Main.java for 5 times and record the performance (in `ms`) in the following table:

	Regular Linked List		Difference List	
Trial	Left Associated	Right Associated	Left Associated	Right Associated
1	20	7	40	13
2	21	8	43	15
3	26	7	42	13
4	24	7	39	14
5	25	7	35	12
Average	23.2	7.2	39.8	13.4