Page 426. The code given in Example 68 does not implement the order defined mathematically in that example. Indeed, defining a procedure that decides whether one List is less than or equal to another is non-trivial and requires some trickery unless one first reverses the lists.

In order to implement this we need to realize that what we have to do is best thought of as a two-step process:

- Reverse the two lists.
- Check whether the reversal of the first list is a prefix of the reversal of the second.

These two can be implemented without much problem.

Reversing a list can be done in analogy with Example 6.12.

```java
public static List reverse (List l)
{
    if (l == null)
        return l;
    else
        return concat(reverse(l.next), new List (l.value, null));
}
```

Checking whether one list is a prefix of another is not so difficult:

```java
public static boolean isPrefixOf (List l1, List l2)
{
    if (l1 == null)
        return true;
    else {
        if (l2 == null)
            return false;
        else {
            if (l1.value == l2.value)
                return isPrefixOf (l1.next,l2.next);
            else
                return isPrefixOf (l1.next,l2.next);
        }
    }
```
public static boolean lessthan (List l1, List l2)
{
    List r1 = reverse (l1);
    List r2 = reverse (l2);
    return isPrefixOf(r1, r2);
}

You may want to have a go at writing a procedure which does not require reversing the two lists first.