

Exercises

The following is a fragment of a class that maintains a singly linked list of Comparables.

```
public class SLL {
    /*
     * Objects of this class are nodes in our singly linked list.
     */
    private static class SLLNode {
        private Comparable data; // the data stored in this node
        private SLLNode next; // the link to the next node
        /*
         * Constructor for an SLLNode with data and next fields as given.
         */
        private SLLNode(Comparable data, SLLNode next) {
            this.data = data;
            this.next = next;
        }
    }

    private SLLNode head = null; // head of the list, or null
}
```

Write the following methods. Method names are up to you. Parameter and return types can be deduced from the descriptions.

1. Write a method that returns (but does not remove) a maximum element in the list, as determined by the `compareTo` method. If the list is empty, your method should return `null`.
2. Write a method that returns (but does not remove) a minimum element in the list, as determined by the `compareTo` method. If the list is empty, your method should return `null`.
3. Write a method that removes and returns a maximum element in the list, as determined by the `compareTo` method. If the list is empty, your method should return `null`.
4. Write a method that removes and returns a minimum element in the list, as determined by the `compareTo` method. If the list is empty, your method should return `null`.
5. Write a method that tests whether the list is in non-decreasing order. (An empty list is.)
6. Write a method that tests whether the list is in strictly decreasing order. (An empty list is.)
7. Write a method that tests whether the list contains any duplicate elements. You may assume that the list is in non-decreasing order.

Bring your solutions in hard copy (printed or handwritten) to class on Friday, April 21.