First order of business: bug check.
Write a constructor for `LinkedIntList` that accepts an int parameter and makes a list of the number from 0 to n.

- new `LinkedIntList(3)`: 

```
front = data | next
---------|---------
3        |        |
2        |        |
1        |        |
0        |        |
```
**addSorted**

- Write a method `addSorted` that accepts an `int` as a parameter and adds it to a sorted list in sorted order.

  - **Before `addSorted(17)`**:

    `front = data next`

    ![Element diagram before adding 17]

    element 0: data = -4, next = 8
    element 1: data = 8, next = 22
    element 2: data = 22

  - **After `addSorted(17)`**:

    `front = data next`

    ![Element diagram after adding 17]

    element 0: data = -4, next = 8
    element 1: data = 8, next = 17
    element 2: data = 17, next = 22
    element 3: data = 22
changing a list

• There are only two ways to change a linked list:
  • Change the value of `front` (modify the front of the list)
  • Change the value of `<node>.next` (modify middle or end of list to point somewhere else)

• Implications:
  • To add in the middle, need a reference to the `previous` node
  • Front is often a special case
Common cases

- **middle**: "typical" case in the middle of an existing list
- **back**: special case at the back of an existing list
- **front**: special case at the front of an existing list
- **empty**: special case of an empty list