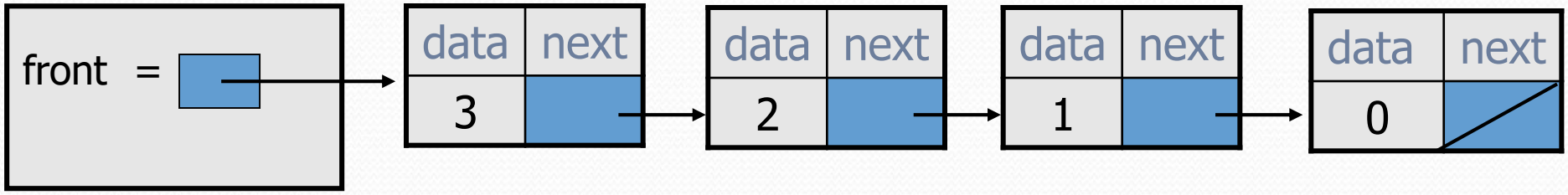

First order of business : bug check.

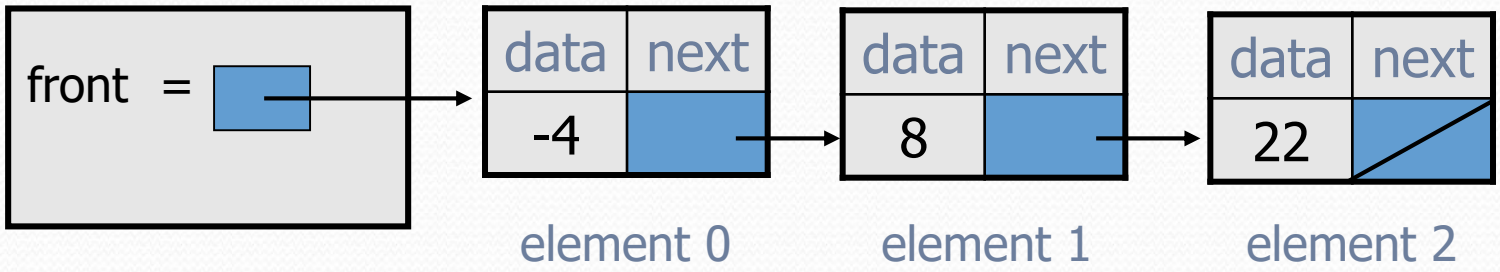
LinkedIntList(int n)

- Write a constructor for `LinkedIntList` that accepts an `int n` parameter and makes a list of the number from 0 to `n`
  - `new LinkedIntList(3)` :

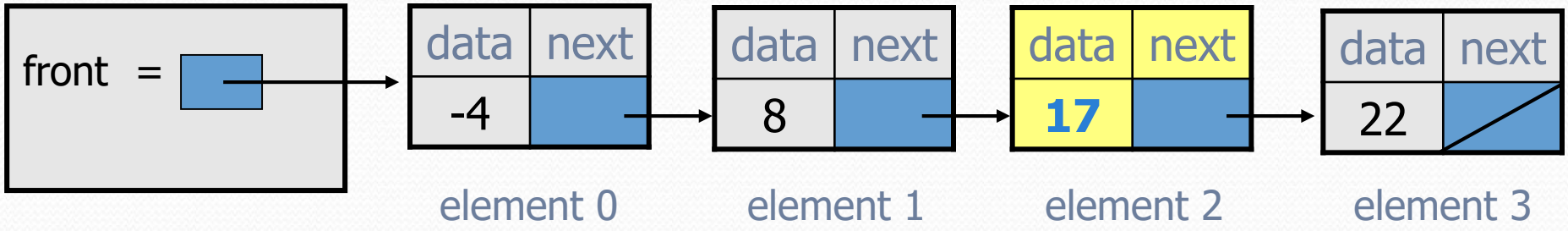


addSorted

- Write a method `addSorted` that accepts an `int` as a parameter and adds it to a sorted list in sorted order.
  - Before `addSorted(17)` :



  - After `addSorted(17)` :



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


INFINITE