1. No answers were required here.

2. Here is what things look like when the code is done:

3. Here are the answers, expressed as a Java comment:

```java
// Returns the object in the middle node in the linked list whose first
// node is referred to by front. If the list has k nodes, the middle
// node is defined to be (k/2)+1. For example, if there are 6 nodes,
// the 4th node is the middle node.
// Preconditions:
// - front refers to the first node in a linked list containing
//   one or more nodes.
// - the number of nodes in the list is even.

public static int countOccurrences(ListNode front, Object o) {
    // So far, we haven't seen any occurrences.
    int count = 0;

    // Loop through every node of the linked list.
    while (front != null) {
        // If this node contains an object that equals o, count it.
        if (front.data.equals(o))
            count++;
        front = front.link;
    }

    // Return the final count.
    return count;
}
```

The first sentence gives the general idea, but we are careful to add a sentence saying exactly what we mean by “middle”. (For lists of even length, there is no exact middle element!)

Make sure you understand what would go wrong if either of the preconditions were violated. Would the method crash (if so, why?) or would it fail to meet its promise?

4. Here is the completed method: