Problem 1

```
public static IntNode addBefore(IntNode front, int target, int newItem) {

    IntNode previous = null;
    IntNode current = front;

    if(front == null)
        return front;

    while(current.data!=target && current!=null){
        previous = front;
        current = current.next;
    }

    IntNode insertNode = new IntNode(newItem, current);

    previous.next = insertNode;

    if (previous == null)
       return insertNode;

    return front;
}
```

Problem 2

```
public static IntNode addBeforeLast(IntNode front, int item){

    IntNode current = front;
    IntNode previous = null;

    if(front == null)
        return null;

    while(current.next!=null){
        previous = current;
        current = current.next;
    }

    IntNode insertNode = new IntNode (item, current);

    if (previous == null){
         return insertNode;
    }

    previous.next = insertNode;

    return front;
}
```

Problem 3

```
public static int numberOfOccurrences(StringNode front, String target) {

    int counter = 0;

    StringNode current = front;

    while(current!=null){
        if(current.data.equals(target))
            counter++;
        current = current.next;
    }

    return counter;
}
```

Problem 4

```
public static void deleteEveryOther(IntNode front){

    if(front == null)
        return;

    boolean everyOther = true;

    IntNode previous = front;
    IntNode current = front.next;

    while(current!=null){
      if(everyOther){
          previous=current;
          current = current.next;
      }
      else{
        current = current.next;
        previous.next = current;
      }

      everyOther = !(everyOther);
    }
}
```

Problem 5

```
public static StringNode deleteAllOccurrences(StringNode front, String target){

    if(front == null)
        return null;

    StringNode previous = null;
    StringNode current = front;

    while(current!=null){
        if(current.data.equals(target)){
            if(previous == null){
                front = current.next;
                current = current.next;
            }
            else{
                previous.next = current.next;
                current = current.next;
            }
        }
        else{
            previous = current;
            current = current.next;
        }
    }

    return front;
}
```

Problem 6

```
public IntNode commonElements(IntNode frontL1, IntNode frontL2){

    IntNode first = null;
    IntNode last = null;

    while(frontL1 != null && frontL2 != null){
        if(frontL1.data < frontL2.data)
            frontL1 = frontL1.next;
        else if(frontL2.data > frontL1.data)
            frontL2 = frontL2.next;
        else{
            IntNode current = new IntNode(frontL1.data, null);
            if(last != null)
                last.next = current;
            else
                first = current;

            last = current;
            frontL1 = frontL1.next;
            frontL2 = frontL2.next;
        }
    }

    return first;
}
```