1)

```
*
+*+
----*----
+++*+++
-+---*---+-
```

2)

```
public ArrayIntList runningTotal() {
    ArrayIntList newList = new ArrayIntList(this.size);
    int sum = 0;
    for (int i = 0; i < this.size; i++) {
        sum += this.data[i];
        newList.data[i] = sum;
        newList.size++;
    }
    return newList;
}
```

3)

1)

```
p = new ListNode(2, p);
```

2)

```
p.next = q.next;
q.next = null;
```

3)

```
ListNode temp = q;
q = q.next;
temp.next = p.next;
p.next = temp;
```

4)

```
temp = p.next;
temp.next.next = p;
p = temp.next;
p.next.next = null;
temp.next = q;
q = temp;
```

4)

```
public static Map<String, Set<String>> convertNames(List<String> names) {
    Map<String, Set<String>> result = new TreeMap<String, Set<String>>();
    for (String name : names) {
        int commaIndex = name.indexOf(",");
        String firstName = name.substring(commaIndex + 2);
        String lastName = name.substring(0, commaIndex);
        if (!result.containsKey(firstName)) {
            result.put(firstName, new TreeSet<String>());
        }
        result.get(firstName).add(lastName));
    }
```

```
        return result;
    }
```

5)

```
    public static String commonChars(String s1, String s2) {
        if (s1.length() != s2.length()) {
            throw new IllegalArgumentException();
        } else if (s1.length() == 0) {
            return "";
        } else {
            if (s1.charAt(0) == s2.charAt(0)) {
                return s1.charAt(0) + commonChars(s1.substring(1), s2.substring(1));
            } else {
                return "." + commonChars(s1.substring(1), s2.substring(1));
            }
        }
    }
```

6)

```
    public static boolean isSortedTriples(Stack<Integer> s) {
        Queue<Integer> q = new LinkedList<Integer>();
        boolean flag = true;
        while (s.size() >= 3) {
            int first = s.pop();
            q.add(first);
            int second = s.pop();
            q.add(second);
            int third = s.pop();
            q.add(third);
            if (first > second || second > third) {
                flag = false;
            }
        }
        while (!s.isEmpty()) {
            q.add(s.pop());
        }
        while (!q.isEmpty()) {
            s.push(q.remove());
        }
        while (!s.isEmpty()) {
            q.add(s.pop());
        }
        while (!q.isEmpty()) {
            s.push(q.remove());
        }
        return flag;
    }
```