CSE 143, Winter 2009
Sample Midterm Exam #2 Key

1.

```
public void interleave(Queue<Integer> q) {
    if (q.size() % 2 != 0) {
        throw new IllegalArgumentException();
    }
    Stack<Integer> s = new Stack<Integer>();
    int halfSize = q.size() / 2;
    for (int i = 0; i < halfSize; i++) {
        s.push(q.remove());
    }
    while (!s.isEmpty()) {   // s2q(s, q);
        q.add(s.pop());
    }
    for (int i = 0; i < halfSize; i++) {
        q.add(q.remove());
    }
    for (int i = 0; i < halfSize; i++) {
        s.push(q.remove());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
        q.add(q.remove());
    }
}
```

2. Two solutions are shown.

```
public Map<Integer, Integer> union(Map<Integer, Integer> m1,
                                   Map<Integer, Integer> m2) {
    Map<Integer, Integer> result = new TreeMap<Integer, Integer>();
    for (int key : m1.keySet()) {
        result.put(key, m1.get(key));
    }
    for (int key : m2.keySet()) {
        if (result.containsKey(key)) {
            result.put(key, result.get(key) + m2.get(key));
        } else {
            result.put(key, m2.get(key));
        }
    }
    return result;
}

public Map<Integer, Integer> union(Map<Integer, Integer> m1,
                                   Map<Integer, Integer> m2) {
    Map<Integer, Integer> result = new TreeMap<Integer, Integer>();
    for (int key : m1.keySet()) {
        int value = m1.get(key);
        if (m2.containsKey(key)) {
            int value2 = m2.get(key);
            value += value2;
        }
        result.put(key, value);
    }
    for (int key : m2.keySet()) {
        int value = m2.get(key);
        if (!result.containsKey(key)) {
            result.put(key, value);
        }
    }
    return result;
}
```

3.

```
ListNode list2 = list.next.next;          // list2 -> 3
list.next.next.next.next = list.next;    // 4 -> 2
ListNode temp = list;                     // temp -> 1
list = list.next.next.next;               // list -> 4
list2.next = temp;                        // 3 -> 1
list.next.next = null;                    // 2 /
list2.next.next = null;                   // 1 /
```

4.

```
public void removeLast(int n) {
    if (front != null) {
        ListNode current = front;
        ListNode spot = null;
        while (current.next != null) {
            if (current.next.data == n) {
                spot = current;
            }
            current = current.next;
        }
        if (spot != null) {
            spot.next = spot.next.next;
        } else if (front.data == n) {
            front = front.next;
        }
    }
}
```

5.

Call	Output
mystery(3, 3);	=3=
mystery(5, 1);	*
mystery(1, 5);	5 4 =3= 2 1
mystery(2, 7);	7 6 5 * 4 3 2
mystery(1, 8);	8 7 6 5 * 4 3 2 1

6.

```
public String repeat(String s, int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
    } else if (n == 0) {
        return "";
    } else if (n == 1) {
        return s;
    } else if (n % 2 == 0) {
        String temp = repeat(s, n / 2);
        return temp + temp;
        // alternative without temp: return repeat(s + s, n / 2);
    } else {
        return s + repeat(s, n - 1);
    }
}
```