## ANSWER KEY

<table>
<thead>
<tr>
<th>SECTION 1</th>
<th>SECTION 2</th>
<th>SECTION 3 (TRACERY)</th>
<th>SECTION 4 (RECURSION)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 B D E</td>
<td>2.1 G</td>
<td>3.1 Boo has a score of 0</td>
<td>4.1 NONE</td>
</tr>
<tr>
<td>1.2 A C</td>
<td>2.2 D</td>
<td>Joe has a score of 0</td>
<td>4.2 E</td>
</tr>
<tr>
<td>1.3 D</td>
<td>2.3 C</td>
<td>Mike has a score of 0</td>
<td>4.3 A</td>
</tr>
<tr>
<td>1.4 C</td>
<td>2.4 I</td>
<td>Sully has a score of 0</td>
<td>4.4 D</td>
</tr>
<tr>
<td>1.5 A B</td>
<td>2.5 G</td>
<td>Lily has a score of 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.6 E</td>
<td>3.2 Lily has a score of 24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Joe has a score of 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Boo has a score of 9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mike has a score of 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sully has a score of 8</td>
<td></td>
</tr>
</tbody>
</table>

## SECTION 5 (DEBUGGING)

<table>
<thead>
<tr>
<th>Line</th>
<th>Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>07</td>
<td>Change public to private</td>
</tr>
<tr>
<td>33</td>
<td>Change &gt; to &gt;=</td>
</tr>
<tr>
<td>36/37</td>
<td>Insert numGuests++;</td>
</tr>
<tr>
<td>41</td>
<td>Change true to false</td>
</tr>
<tr>
<td>42</td>
<td>Change == to !=</td>
</tr>
<tr>
<td>43</td>
<td>Change == name to .equals(name)</td>
</tr>
</tbody>
</table>
SECTION 6 (Linked Lists)

6.1
//Constructor to create a list with one element -- first
public Garden(Flowe first) {
    if (first != null) {
        this.first = first;
        size = 1;
    }
}

6.2
public void add(Flowe f) {
    //error checking
    if (f == null) { return; }
    if (first == null) {
        first = f;
        size++;
    } else {
        // walk to end of list
        Flower current = first;
        while (current.next != null) {
            current = current.next;
        }
        // insert f at tail
        current.next = f;
        f.next = null;
        size++;
    }
}

public void add(Flowe f) {
    if (f == null) { return; }
    if (first == null) {
        first = f;
        size++;
    } else {
        // walk to end of list
        Flower current = first;
        for (; current.next != null; current = current.next) {}
        // insert f at tail
        current.next = f;
        f.next = null;
        size++;
    }
}
6.3.)

// This method identifies which two Flowers in the Garden have
// the smallest combined sum, deletes them, and inserts a new Flower
// at the head of the list that has that sum.

public void deleteSmallestSum() {
    if (size < 2) { return; }

    //pointers
    int smallestSize = Integer.MAX_VALUE;
    Flower smallestOne = null;
    Flower currentPrev = first;
    Flower smallestPrev = first;

    for (Flower current = first; current.next != null; current = current.next) {
        if (current.size + current.next.size < smallestSize) {
            smallestOne = current;
            smallestSize = smallestOne.size + smallestOne.next.size;
            smallestPrev = currentPrev;
        }
        currentPrev = current;
    }

    smallestPrev.next = smallestOne.next.next;
    smallestOne.size = smallestSize;
    size = size - 2;

    add(smallestOne);
}
public class ArraySet implements Set {
    private String[] arr;
    private int size = 0;

    // 2 pts total
    public ArraySet(int capacity) {
        arr = new String[capacity];
    }

    // 5 pts total
    public void add(String s) {
        if (s == null)
            throw new IllegalArgumentException();
        if (contains(s)) return;
        if (size() == arr.length)
            throw new RuntimeException();
        arr[size++] = s;
    }

    // 3 points total
    public boolean contains(String s) {
        for (int i = 0; i < size; i++) {
            if (arr[i].equals(s))
                return true;
        }
        return false;
    }

    // 2 points total
    public int size() {
        return size;
    }

    // 2 points
    public void printSet() {
        for (int i = 0; i < size; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}