ANSWER KEY

SECTION 1
1.1 B D E
1.2 A C
1.3 D
1.4 C
1.5 A B

SECTION 2
2.1 G
2.2 D
2.3 C
2.4 I
2.5 G
2.6 E

SECTION 3 (TRACERY)
3.1
Boo has a score of 0
Joe has a score of 0
Mike has a score of 0
Sully has a score of 0
Lily has a score of 0

3.2
Lily has a score of 24
Joe has a score of 16
Boo has a score of 9
Mike has a score of 8
Sully has a score of 8

SECTION 4 (RECURSION)
4.1 NONE
4.2 E
4.3 A
4.4 D

SECTION 5 (DEBUGGING)

Line	Correction
07	Change `public` to `private`
33	Change > to >=
36/37	Insert `numGuests++;`
41	Change `true` to `false`
42	Change == to `!=`
43	Change `== name` to `.equals(name)`

SECTION 6 (Linked Lists)

6.1

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

6.2

```
    public void add(Flower 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(Flower 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);
    }
}
```

SECTION 7 (DATA STRUCTURES)

```
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 point
    public int size() {
       return size;
    }

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