Key to CSE142 Sample Final Exam, Spring 2018

1. Solution:

```
44,22 [44, 77] 0
44,22 [44, 77] 55
22,22 77
```

2.

Original Array	Final Array
{1, 1, 3}	[1, 2, 3]
{2, 1, 2, 4}	[2, 2, 3, 4]
{6, 13, 0, 3, 7}	[6, 3, 3, 5, 7]
{-1, 6, 3, 5, -3}	[-1, 1, 3, 0, -3]
{7, 2, 3, 1, -3, 12}	[7, 5, 3, 0, 6, 12]

3. Solution:

```
Diamond
Ruby 1
Emerald 2

Emerald
Ruby 1
Emerald 2

Emerald
Emerald 1
Emerald 2

Garnet
Garnet 1
Emerald 2
```

4. Two possible solutions appear below.

```
public static int censorNames(Scanner fileScan) {
    int names = 0;

    while (fileScan.hasNext()) {
        names++;

        System.out.print(fileScan.next() + " ");
        String lname = fileScan.next();
        for (int i = 0; i < lname.length(); i++) {
            System.out.print("X");
        }
        System.out.println();
    }
    return names;
}
```

```
public static int censorNames(Scanner fileScan) {
    int names = 0;
    boolean first = true;

    while (fileScan.hasNext()) {
        String token = fileScan.next();
        if (first) {
            System.out.print(token + " ");
            names++;
        } else {
            for (int i = 0; i < token.length(); i++) {
                System.out.print("X");
            }
            System.out.println();
        }
        first = !first;
    }
    return names;
}
```

5. One possible solution appears below.

```
public static int analyzeParagraphs(Scanner input) {
    int max = 0;
    while (input.hasNextLine()) {
        String line = input.nextLine();
        int count = 0;
        while (!line.equals("<p>")) {
            count++;
            line = input.nextLine();
        }
        System.out.println(count + "-line paragraph");
        if (count > max) {
            max = count;
        }
    }
    return max;
}
```

6. Two possible solutions appear below.

```
public static double[] getTotalValues(String sequence, double[] weights) {
   String nucs = "ACGT";
   double[] result = new double[4];

   int currWeight = 0;
   for (int i = 0; i < sequence.length(); i++) {
      int index = nucs.indexOf(sequence.charAt(i));
      if (index >= 0) {
         result[index] += weights[currWeight];
         currWeight++;
      }
   }

   return result;
}
```

```
public static double[] getTotalValues(String sequence, double[] weights) {
   double[] result = new double[4];

   String noJunk = sequence.replace("-", "");
   for (int i = 0; i < noJunk.length(); i++) {
      int index = -1;
      if (noJunk.charAt(i) == 'A') {
         index = 0;
      } else if (noJunk.charAt(i) == 'C') {
         index = 1;
      } else if (noJunk.charAt(i) == 'G') {
         index = 2;
      } else if (noJunk.charAt(i) == 'T') {
         index = 3;
      }
      if (index >= 0) {
         result[index] += weights[i];
      }
   }

   return result;
}
```

7. Three possible solutions appear below.

```
public static void split(ArrayList<Integer> list) {
    for (int i = 0; i < list.size(); i += 2) {
        int n = list.get(i);
        list.set(i, n / 2 + n % 2);
        list.add(i + 1, n / 2);
    }
}
```

```
public static void split(ArrayList<Integer> list) {

    for (int i = 0; i < list.size(); i += 2) {
        int n = list.remove(i);
        list.add(i, n / 2 + n % 2);
        list.add(i + 1, n / 2);
    }
}
```

```
public static void split(ArrayList<Integer> list) {
    int index = 0;
    while (index < list.size()) {
        int n = list.get(i);
        if (n % 2 == 0) {
            list.add(index, n / 2);
            list.add(index, n / 2);
            list.remove(index + 2);
        } else {
            list.add(index, n / 2);
            list.add(index, n / 2 + 1);
            list.remove(index + 2);
        }
        index += 2;
    }
}
```

8. One possible solution appears below.

```
public class Panther extends Critter {
    private boolean hunting;
    private Random rand;

    public Panther() {
        hunting = false;
        rand = new Random();
    }

    public Color getColor() {
        if (hunting) {
            return Color.RED;
        } else {
            return Color.BLACK;
        }
    }

    public boolean eat() {
        boolean shouldEat = !hunting;
        hunting = true;
        return shouldEat;
    }

    public Attack fight(String opponent) {
        if (hunting) {
            hunting = false;
            return Attack.SCRATCH;
        } else {
            return Attack.ROAR;
        }
    }

    public Direction getMove() {
        int check = rand.nextInt(4);
        if (check == 0) {
            return Direction.NORTH;
        } else if (check == 1) {
            return Direction.EAST;
        } else if (check == 2) {
            return Direction.SOUTH;
        } else { // check == 3
            return Direction.WEST;
        }
    }
}
```

9. Three possible solutions appear below.

```
public static int[] insertMiddle(int[] a, int[] b) {
    int[] result = new int[a.length + b.length];
    for (int i = 0; i < a.length / 2; i++) {
        result[i] = a[i];
    }

    for (int i = 0; i < b.length; i++) {
        result[i + a.length / 2] = b[i];
    }

    for (int i = a.length / 2; i < a.length; i++) {
        result [i + b.length] = a[i];
    }

    return result;
}

public static int[] insertMiddle(int[] a, int[] b) {
    int lengthA = a.length;
    int lengthB = b.length;
    int[] result = new int[lengthA + lengthB];
    for (int i = 0; i < result.length; i++) {
        if (i < lengthA / 2)
            result[i] = a[i];
        else if (i < (lengthA / 2 + lengthB))
            result[i] = b[i - lengthA / 2];
        else
            result[i] = a[i - lengthB];
    }
    return result;
}

public static int[] insertMiddle(int[] a, int[] b) {
    int[] c = new int[a.length + b.length];
    int n = 0;

    for (int i = 0; i < a.length / 2; i++) {
        c[n] = a[i];
        n++;
    }

    for (int i = 0; i < b.length; i++) {
        c[n] = b[i];
        n++;
    }

    for (int i = a.length / 2; i < a.length; i++) {
        c[n] = a[i];
        n++;
    }
    return c;
}
```

10. Three possible solutions appear below.

```
public static boolean samePattern(String s1, String s2) {
    if (s1.length() != s2.length()) {
        return false;
    }
    for (int i = 0; i < s1.length(); i++) {
        for (int j = i + 1; j < s1.length(); j++) {
            if (s1.charAt(i) == s1.charAt(j) &&
            s2.charAt(i) != s2.charAt(j)) {
                return false;
            }
            if (s2.charAt(i) == s2.charAt(j) &&
            s1.charAt(i) != s1.charAt(j)) {
                return false;
            }
        }
    }
    return true;
}

public static boolean samePattern(String s1, String s2) {
    if (s1.length() != s2.length()) {
        return false;
    }
    for (int i = 0; i < s1.length(); i++) {
        if (s1.indexOf(s1.charAt(i)) != s2.indexOf(s2.charAt(i))) {
            return false;
        }
    }
    return true;
}

public static boolean samePattern(String s1, String s2) {
    if (s1.length() != s2.length()) {
        return false;
    }
    String chars1 = "";
    String chars2 = "";
    for (int i = 0; i < s1.length(); i++) {
        char ch1 = s1.charAt(i);
        char ch2 = s2.charAt(i);
        if (chars1.indexOf(ch1) != -1) {
            if (chars1.indexOf(ch1) != chars2.indexOf(ch2)) {
                return false;
            }
        } else {
            if (chars2.indexOf(ch2) != -1) {
                return false;
            }
            chars1 += ch1;
            chars2 += ch2;
        }
    }
    return true;
}
```