CSE 142, Spring 2009, Sample Final Exam 1, Sample Solutions

1. Array Simulation

```
{1, 3, 3}


{8, 7, 5, 4, 6}


{1, 2, 2, 2, 2, 1}


{40, 45, 35, 20, 15, 30}


{15, 8, 2, 3, 11, 3, 5, 4}
```

2. Inheritance

```
biden
palin-R
biden-D   palin-R   palin-D

palin
palin-R
palin-R   palin-D

biden
mccain-R
biden-D   mccain-R   palin-D

palin
palin-R   obama-R
palin-R   obama-R   palin-D
```

3. Parameters and References

```
Line 1: 14 14
Line 2: 7 9 14 2
Line 3: 18 18
Line 4: 7 9 14 18
```

4. Token-Based File Processing. One possible solution appears below.

```
        public static void printStrings(Scanner input) {
            while (input.hasNextInt()) {
                int times = input.nextInt();
                String word = input.next();
                for (int i = 0; i < times; i++) {
                    System.out.print(word);
                }
                System.out.println();
            }
        }
```

5. Line-Based File Processing. Two possible solutions appear below.

```
public static void printDuplicates(Scanner input) {
    while (input.hasNextLine()) {
        String line = input.nextLine();
        Scanner lineScan = new Scanner(line);

        String token = lineScan.next();
        int count = 1;

        while (lineScan.hasNext()) {
            String token2 = lineScan.next();
            if (token2.equals(token)) {
                count++;
            } else {
                if (count > 1) {
                    System.out.print(token + "*" + count + " ");
                }
                token = token2;
                count = 1;
            }
        }

        if (count > 1) {
            System.out.print(token + "*" + count);
        }
        System.out.println();
    }
}

public static void printDuplicates(Scanner input) {
    while (input.hasNextLine()) {
        String line = input.nextLine();
        Scanner lineScan = new Scanner(line);

        String token = lineScan.next();
        int count = 1;

        while (lineScan.hasNext()) {
            String token2 = lineScan.next();
            if (token2.equals(token)) {
                count++;
            }

            if (count > 1 && (!lineScan.hasNext() ||
!token2.equals(token))) {
                System.out.print(token + "*" + count + " ");
                count = 1;
            }
            token = token2;
        }

        System.out.println();
    }
}
```

6. Arrays. Five possible solutions appear below.

```
public static double[] arraySum(double[] a1, double[] a2) {
    double[] a3 = new double[Math.max(a1.length, a2.length)];
    for (int i = 0; i < a3.length; i++) {
        if (i >= a1.length) {           // done with a1; take from a2
            a3[i] = a2[i];
        } else if (i >= a2.length) {    // done with a2; take from a1
            a3[i] = a1[i];
        } else {
            a3[i] = a1[i] + a2[i];      // take sum of a1 and a2
        }
    }
    return a3;
}

public static double[] arraySum(double[] a1, double[] a2) {
    double[] a3 = new double[Math.max(a1.length, a2.length)];
    for (int i = 0; i < a1.length; i++) {    // add a1 into result
        a3[i] += a1[i];
    }
    for (int i = 0; i < a2.length; i++) {    // add a2 into result
        a3[i] += a2[i];
    }
    return a3;
}

public static double[] arraySum(double[] a1, double[] a2) {
    double[] a3;                                // create result array
    if (a1.length > a2.length) {
        a3 = new double[a1.length];
    } else {
        a3 = new double[a2.length];
    }
    for (int i = 0; i < a1.length; i++) {      // add a1 into result
        a3[i] += a1[i];
    }
    for (int i = 0; i < a2.length; i++) {      // add a2 into result
        a3[i] += a2[i];
    }
    return a3;
}

public static double[] arraySum(double[] a1, double[] a2) {
    int minLength = Math.min(a1.length, a2.length);
    int maxLength = Math.max(a1.length, a2.length);
    double[] a3 = new double[maxLength];            // create result
array
```

```
    for (int i = 0; i < minLength; i++) {
        a3[i] = a1[i] + a2[i];
    }
    for (int i = minLength; i < maxLength; i++) {   // add a1,a2 into
result
        if (a1.length > a2.length) {
            a3[i] = a1[i];
        } else {
            a3[i] = a2[i];
        }
    }
    return a3;
}

public static double[] arraySum(double[] a1, double[] a2) {
    double[] shorter = a1;
    double[] longer = a2;
    if (a1.length > a2.length) {
        shorter = a2;
        longer = a1;
    }
    double[] a3 = new double[longer.length];
    for (int i = 0; i < shorter.length; i++) {
        a3[i] = shorter[i] + longer[i];
    }
    for (int i = shorter.length; i < longer.length; i++) {
        a3[i] += longer[i];
    }

    return a3;
}
```

7. ArrayList. Two possible solutions appear below.

```
      public static void reverse3(ArrayList<Integer> list) {
          for (int i = 0; i < list.size() - 2; i += 3) {
            int n1 = list.get(i);
            int n3 = list.get(i + 2);
            list.set(i, n3);
            list.set(i + 2, n1);
          }
      }

      public static void reverse3(ArrayList<Integer> list) {
          for (int i = 0; i < list.size() - 2; i += 3) {
            list.add(i, list.remove(i + 2));
            list.add(i + 2, list.remove(i + 1));
          }
      }
```

8. Critters. One possible solution appears below.

```
        public class Orca extends Critter {
```

```
        int count;

        public Orca() {
       count = 0;
        }

        public Action getMove(CritterInfo info) {
       if (count % 6 == 4 || count % 6 == 5) {
           count++;
           return Action.LEFT;
       } else if (info.getFront() != Neighbor.EMPTY) {
           return Action.INFECT;
       } else {
           count++;
           return Action.HOP;
       }
        }

        public String toString() {
       if (count % 6 < 4) {
           return "M";
       } else {
           return "T";
       }
        }
    }
```

9. Arrays. Six possible solutions appear below.

```
public static void partition(int[] a, int v) {
    int i2 = a.length - 1;
    for (int i1 = 0; i1 < i2; i1++) {
        while (i2 > i1 && a[i2] >= v) {
            i2--;
        }
        int temp = a[i1];
        a[i1] = a[i2];
        a[i2] = temp;
    }
}

public static void partition(int[] a, int v) {
    int i1 = 0;
    int i2 = a.length - 1;
    while (true) {
        while (i2 > i1 && a[i2] >= v) {
            i2--;
        }
        while (i2 > i1 && a[i1] <= v) {
            i1++;
        }
        if (i1 >= i2) {
            break;
        }
```

```
            int temp = a[i1];
            a[i1] = a[i2];
            a[i2] = temp;
        }
    }

    public static void partition(int[] a, int v) {
        int[] copy = new int[a.length];
        int target = 0;

        for (int i = 0; i < a.length; i++) {
            if (a[i] < v) {
                copy[target] = a[i];
                target++;
            }
        }

        for (int i = 0; i < a.length; i++) {
            if (a[i] > v) {
                copy[target] = a[i];
                target++;
            }
        }

        for (int i = 0; i < a.length; i++) {
            a[i] = copy[i];
        }
    }

    public static void partition(int[] a, int v) {
        for (int i = 0; i < a.length; i++) {
            int smallest = i;
            for (int j = i + 1; j < a.length; j++) {
                if (a[j] < a[smallest]) {
                    smallest = j;
                }
            }
            int temp = a[i];
            a[i] = a[smallest];
            a[smallest] = temp;
        }
    }

    public static void partition(int[] a, int v) {
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a.length - 1; j++) {
                if (a[j] > a[j + 1]) {
                    int temp = a[j];
                    a[j + 1] = a[j];
                    a[j] = temp;
                }
            }
        }
```

```
}

public static void partition(int[] a, int v) {
    for (int i = 0; i < a.length; i++) {
        if (a[i] > a[i + 1]) {
            int temp = a[i];
            a[i + 1] = a[i];
            a[i] = temp;
            partition(a, v);
        }
    }
}
```

10. Programming. One possible solution appears below.

```
public static String acronym(String s) {
    boolean inWord = false;
    s = s.toUpperCase();
    String result = "";
    for (int i = 0; i < s.length(); i++) {
      char ch = s.charAt(i);
      if (ch == ' ' || ch == '-') {
          inWord = false;
      } else if (!inWord) {
          inWord = true;
          result += ch;
      }
    }
    return result;
}
```