CSE 142 Winter 2007
Final Exam Key

1. Expressions

```
2.0
false
1
"33 + 45(int) 6.0"
false
```

2. Array Mystery

```
{42}
{8, 11, 3}
{14, 19, 25, 6}
{1, 2, 2, 3, 3, 0}
{10, 14, 16, 22, 30, 31, 1}
```

3. Inheritance Mystery

```
I'm Batman.
Batman A
Hulk B

Up up and away!
WonderWoman A
Superman B

Hulk smash!
Hulk A
Hulk B

Up up and away!
Batman A
Superman B
```

4. File Processing

Two solutions are shown.

```
public static int count(Scanner input, String word) {
    int count = 0;
    while (input.hasNext()) {
        String nextWord = input.next();
        nextWord = nextWord.toLowerCase();
        if (nextWord.equals(word)) {
            count++;
        }
    }
    return count;
}

public static int count(Scanner input, String word) {
    int count = 0;
    while (input.hasNext()) {
        if (input.next().equalsIgnoreCase(word)) {
            count++;
        }
    }
    return count;
}
```

5. File Processing

Two solutions are shown.

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

        double totalBuy = 0.0;
        double totalSell = 0.0;
        while (lineScan.hasNext()) {
            String stockName = lineScan.next();
            double buy  = lineScan.nextDouble();
            double sell = lineScan.nextDouble();
            totalBuy += buy;
            totalSell += sell;
        }
        double totalProfit = totalSell - totalBuy;
        System.out.println(name + ": total profit = $" + totalProfit);
    }
}
```

```
public static void analyzeStocks(Scanner input) {
    while (input.hasNextLine()) {
        Scanner lineScan = new Scanner(input.nextLine());
        String name = lineScan.next();
        double profit = 0.0;
        while (lineScan.hasNext()) {
            lineScan.next();  // throw away stock name
            profit = profit - lineScan.nextDouble() + lineScan.nextDouble();
        }
        System.out.println(name + ": total profit = $" + profit);
    }
}
```

6. Arrays

Three solutions are shown.

```
public static boolean isReverse(int[] a1, int[] a2) {
    if (a1.length != a2.length) {
        return false;
    }

    for (int i = 0; i < a1.length; i++) {
        if (a1[i] != a2[a2.length - 1 - i]) {
            return false;
        }
    }
    return true;
}

public static boolean isReverse(int[] a1, int[] a2) {
    if (a1.length != a2.length) {
        return false;
    }

    int count = 0;
    for (int i = 0; i < a1.length; i++) {
        if (a1[i] != a2[a2.length - 1 - i]) {
            count++;
        }
    }

    return count == a1.length;
}

public static boolean isReverse(int[] a1, int[] a2) {
    if (a1.length == a2.length) {
        int j = a1.length - 1;
        for (int i = 0; i < a1.length; i++) {
            if (a1[i] != a2[j]) {
                return false;
            }
            j--;
        }
        return true;
    } else {
        return false;
    }
}
```

7. Arrays

Four solutions are shown.

```
public static void wrapHalf(int[] a) {
    for (int i = 0; i < a.length / 2; i++) {
        // rotate right
        int last = a[a.length - 1];
        for (int j = a.length - 1; j >= 1; j--) {
            a[j] = a[j - 1];
        }
        a[0] = last;
    }
}
```

```
public static void wrapHalf(int[] a) {
    int[] temp = new int[a.length];
    int mid = (a.length + 1) / 2;
    int rest = a.length - mid;
    for (int i = 0; i < mid; i++) {       // copy first half
        temp[i + rest] = a[i];
    }
    for (int i = 0; i < rest; i++) {      // copy second half
        temp[i] = a[i + mid];
    }
    for (int i = 0; i < a.length; i++) {  // copy back into a
        a[i] = temp[i];
    }
}
```

```
public static void wrapHalf(int[] a) {
    int[] temp = new int[a.length / 2];
    int largerHalf = (a.length + 1) / 2;
    for (int i = 0; i < temp.length; i++) {  // copy right half to temp
        temp[i] = a[i + largerHalf];
    }
    for (int i = a.length - 1; i >= temp.length; i--) {  // shift left half
        a[i] = a[i - temp.length];
    }
    for (int i = 0; i < temp.length; i++) {  // copy left half back to a
        a[i] = temp[i];
    }
}
```

```
public static void wrapHalf(int[] a) {
    int half = a.length / 2;
    for (int i = 0; i < half; i++) {
        if (a.length % 2 == 1) {
            swap(a, i + half, i + half + 1);
        }
        swap(a, i, i + half);
    }
}
private static void swap(int[] a, int i, int j) {
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

8. Critters

Four solutions are shown.

```
public class Baboon implements Critter {
    private int direction;
    private int count;
    private int max;

    public Baboon() {
        direction = NORTH;
        count = 0;
        max = 1;
    }
    ...
    public int getMove(CritterInfo info) {
        if (count >= max) {
            count = 0;
            max++;
            if (max > 8) {
                max = 1;
            }

            if (direction == NORTH) {
                direction = EAST;
            } else if (direction == EAST) {
                direction = SOUTH;
            } else if (direction == SOUTH) {
                direction = WEST;
            } else {  // WEST
                direction = NORTH;
            }
        }

        count++;
        return direction;
    }
}
```

```
public class Baboon implements Critter {
    private int count;
    private int max;

    public Baboon() {
        max = 1;
    }
    ...
    public int getMove(CritterInfo info) {
        count++;
        if (count > max) {
            count = 1;
            max++;
            if (max > 8) {
                max = 1;
            }
        }

        if (max == 1 || max == 5) {
```

```
            return EAST;
        } else if (max == 2 || max == 6) {
            return SOUTH;
        } else if (max == 3 || max == 7) {
            return WEST;
        } else {  // WEST
            return NORTH;
        }
    }
}

public class Baboon implements Critter {
    private int count;
    private int max;

    public Baboon() {
        count = 0;
        max = 1;
    }
    ...
    public int getMove(CritterInfo info) {
        int[] DIRECTIONS = {NORTH, EAST, SOUTH, WEST};
        int direction = DIRECTIONS[(max - 1) % 4];
        if (count >= max) {
            count = 0;
            max++;
        }
        count++;
        return direction;
    }
}

public class Baboon implements Critter {
    private int count;

    public int getMove(CritterInfo info) {
        count++;
        if (count > 36) {
            count = 1;
        }
        if (count <= 1 || (count >= 11 && count <= 15)) {
            return NORTH;
        } else if (count <= 3 || (count >= 16 && count <= 21)) {
            return EAST;
        } else if (count <= 6 || (count >= 22 && count <= 28)) {
            return SOUTH;
        } else { // if (count <= 10 || (count >= 29 && count <= 36)) {
            return WEST;
        }
    }
}
```

9. Classes

Three solutions are shown.

```
public void subtractWeeks(int weeks) {
    for (int i = 1; i <= weeks; i++) {
        day -= 7;
        if (day <= 0) {
            month--;
            if (month == 0) {
                month = 12;
            }
            day += numDays(month);
        }
    }
}
```

```
public void subtractWeeks(int weeks) {
    int days = 7 * weeks;
    for (int i = 1; i <= days; i++) {
        day--;
        if (day == 0) {
            month--;
            if (month == 0) {
                month = 12;
            }
            day = numDays(month);
        }
    }
}
```

```
public void subtractWeeks(int weeks) {
    day -= 7 * weeks;
    while (day <= 0) {
        month--;
        if (month == 0) {
            month = 12;
        }
        day += numDays(month);
    }
}
```