# CSE 142, Autumn 2008
# Final Exam Key

## 1. Expressions

| Expression | Value |
|---|---|
| `-(1 + 2 * 3 + (1 + 2) * 3)` | `-16` |
| `1 + 2 + "(3 + 4)" + 5 * 6 + 7` | `"3(3 + 4)307"` |
| `30 % 9 + 5 % 8 - 11 % 4 % 2` | `7` |
| `4 < 10 != (5 == 6 \|\| 9 >= 9)` | `false` |
| `(double) 45 / 10.0 / 2 + 7 / 4.0 * 10` | `19.75` |

## 2. Array Mystery

| Expression | Final Contents of Array |
|---|---|
| `int[] a1 = {1, 2, 3};`<br>`arrayMystery(a1);` | `{1, 3, 3}` |
| `int[] a2 = {8, 2, 3, 1, 6};`<br>`arrayMystery(a2);` | `{8, 7, 5, 4, 6}` |
| `int[] a3 = {1, 1, 1, 1, 1, 1};`<br>`arrayMystery(a3);` | `{1, 2, 2, 2, 2, 1}` |
| `int[] a4 = {40, 10, 25, 5, 10, 30};`<br>`arrayMystery(a4);` | `{40, 45, 35, 20, 15, 30}` |
| `int[] a5 = {15, 6, -1, 4, 8, -2, 7, 4};`<br>`arrayMystery(a5);` | `{15, 8, 2, 3, 11, 3, 5, 4}` |

## 3. Inheritance Mystery

```
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
```

**4. File Processing**

```
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();
    }
}
```

**5. Array Programming**

```
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;
}
```

**6. Critters**

```
public class Hyena extends Critter {
    private int moves;
    private int width;

    public Hyena() {
        moves = 0;
        width = 1;
    }

    public boolean eat() {
        moves = 0;
        width++;
        return true;
    }

    public Direction getMove() {
        moves++;
        if (moves > 2 * width + 2) {
            moves = 1;
            width++;
        }

        if (moves == 1) {
            return Direction.NORTH;
        } else if (moves <= width + 1) {
            return Direction.EAST;
        } else if (moves == width + 2) {
            return Direction.SOUTH;
        } else {
            return Direction.WEST;
        }
    }
}
```

```
public class Hyena extends Critter {
    private int moves = 0;
    private int max = 1;
    private boolean ate = false;

    public boolean eat() {
        ate = true;
        return true;
    }

    public Direction getMove() {
        moves++;
        if (ate || moves == 2 * max + 3) {
            ate = false;
            moves = 1;
            max++;
        }

        if (moves == 1) {
            return Direction.NORTH;
        } else if (moves <= max + 1) {
            return Direction.EAST;
        } else if (moves == max + 2) {
            return Direction.SOUTH;
        } else {
            return Direction.WEST;
        }
    }
}
```

7. **Objects**

```
public int absoluteDay() {
    int myMonth = month;
    int myDay = day;
    month = 1;
    day = 1;
    int count = 1;
    while (month != myMonth || day != myDay) {
        count++;
        nextDay();
    }
    return count;
}

public int absoluteDay() {
    Date temp = new Date(1, 1);
    int count = 1;
    while (day != temp.day || month != temp.month) {
        count++;
        temp.nextDay();
    }
    return count;
}

public int absoluteDay() {
    int count = day;
    Date temp = new Date(1, 1);
    for (int i = 1; i < month; i++) {
        count += temp.daysInMonth();
        temp.month++;
    }
    return count;
}

public int absoluteDay() {
    int count = 0;
    for (int i = 1; i <= month - 1; i++) {
        if (i == 4 || i == 6 || i == 9 || i == 11) {
            count += 30;
        } else if (i == 2) {
            count += 28;
        } else {
            count += 31;
        }
    }
    count += day;
    return count;
}

public int absoluteDay() {
   int[] dayCount = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
   int count = 0;
   for (int i = 1; i <= month - 1; i++) {
      count += dayCount[i - 1];
   }
   return count + day;
}

public int absoluteDay() {
    Date copy = new Date(month, day);
    int count = 365;
    while (copy.getMonth() != 12 || copy.getDay() != 31) {
        copy.nextDay();
        count--;
    }
    return count;
}
```

```
public int absoluteDay() {
    int oldMonth = month;
    int oldDay = day;
    int count = 0;
    while (month != 12 || day != 31) {
        nextDay();
        count++;
    }
    month = oldMonth;
    day = oldDay;
    return 365 - count;
}

public int absoluteDay() {
    int oldMonth = month;
    month = 0;
    int count = 0;
    for (int i = 1; i < oldMonth; i++) {
        month = i;
        count += daysInMonth();
    }
    count += day;
    return count;
}

public int absoluteDay() {
    int oldMonth = month;
    int count = 0;
    for (int i = month; i > 1; i--) {
        month--;
        count += daysInMonth();
    }
    count += day;
    month = oldMonth;
    return count;
}
```

**8. Array Programming**

```
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);
        }
    }
}
```