CSE 142, Autumn 2009
Midterm Exam Key

1. Expressions

Expression	Value
`1 + 2 * 3 * 4 - 5 * 2`	`15`
`2 + "(int)2.0" + 2 * 2 + 2`	`"2(int)2.042"`
`15 % 3 == 0 && !(3 > 2 && 1 > 3)`	`true`
`1 / 2 + -(157 / 10 / 10.0) + 9.0 * 1 / 2`	`3.0`
`24 % 5 + 9 % (6 % 4)`	`5`

2. Parameter Mystery

```
semi missing a brace and 42
semi missing a 42 and 8
brace missing a literal and semi
84 missing a 1 and cse
```

3. If/Else Simulation

Method Call	Output
`ifElseMystery(2, 10, 3);`	`5 10 4`
`ifElseMystery(8, 6, 1);`	`2 5 1`
`ifElseMystery(4, 6, 7);`	`12 6 9`
`ifElseMystery(20, 5, 5);`	`16 4 6`

4. While Loop Simulation

Method Call	Output
`whileMystery(25, 2);`	`12, 1`
`whileMystery(10345, 10);`	`1034, 103, 10, 3`
`whileMystery(63, 2);`	`31, 15, 7, 3, 1, 0, 6`

5. Assertions

	`n == 0`	`even <= odd`	`n % 2 == 0`
Point A	SOMETIMES	ALWAYS	SOMETIMES
Point B	NEVER	SOMETIMES	ALWAYS
Point C	NEVER	ALWAYS	NEVER
Point D	SOMETIMES	SOMETIMES	SOMETIMES
Point E	SOMETIMES	SOMETIMES	SOMETIMES

6. Programming

There are many ways to solve any programming problem. Here are three common correct solutions we saw:

```
// longer solution; pull first post out of fencepost
public static void longestName(Scanner console, int names) {
    System.out.print("name #1? ");
    String longest = console.next();
    int count = 1;
    for (int i = 2; i <= names; i++) {
        System.out.print("name #" + i + "? ");
        String name = console.next();
        if (name.length() == longest.length()) {
            count++;
        }
        if (name.length() > longest.length()) {
            longest = name;
            count = 1;
        }
    }
    String fixedName = longest.substring(0, 1).toUpperCase();
    fixedName = fixedName + longest.substring(1).toLowerCase();
    System.out.println(fixedName + "'s name is longest");
    if (count > 1) {
        System.out.println("(There was a tie!)");
    }
}

// shorter solution; boolean flag
public static void longestName(Scanner console, int names) {
    String longest = "";
    boolean tie = false;
    for (int i = 1; i <= names; i++) {
        System.out.print("name #" + i + "? ");
        String name = console.next();
        if (name.length() > longest.length()) {
            longest = name;
            tie = false;
        } else if (name.length() == longest.length()) {
            tie = true;
        }
    }
    longest = longest.substring(0, 1).toUpperCase() + longest.substring(1).toLowerCase();
    System.out.println(longest + "'s name is longest");
    if (tie) {
        System.out.println("(There was a tie!)");
    }
}

// store longest two strings and compare
public static void longestName(Scanner console, int n) {
    String longest = "";
    String longest2 = "";
    for (int i = 1; i <= n; i++) {
        System.out.print("name #" + i + "? ");
        String name = console.next();
        if (name.length() > longest.length()) {
            longest = name;
            longest2 = name;
        } else if (name.length() == longest.length()) {
            longest2 = name;
        }
    }
    String capitalized = longest.toUpperCase().charAt(0) + longest.toLowerCase().substring(1);
    System.out.println(capitalized + "'s name is longest");
    if (!longest.equalsIgnoreCase(longest2)) {
        System.out.println("(There was a tie!)");
    }
}
```

7. Programming

```
// count digits, use Math.pow to scale up; use if/else to compare
public static int largerDigits(int a, int b) {
    int digits = 0;
    int result = 0;
    while (a != 0 && b != 0) {
        if (a % 10 > b % 10) {
            result += a % 10 * (int) Math.pow(10, digits);
        } else {
            result += b % 10 * (int) Math.pow(10, digits);
        }
        a = a / 10;
        b = b / 10;
        digits++;
    }
    return result;
}

// cumulative multiplier and Math.max
public static int largerDigits(int a, int b) {
    int multiplier = 1;
    int result = 0;
    while (a != 0 && b != 0) {
        int digit = Math.max(a % 10, b % 10);
        result += digit * multiplier;
        a = a / 10;
        b = b / 10;
        multiplier = multiplier * 10;
    }
    return result;
}
```

```
public static boolean containsBothDigits(int a, int b, int c) {
    int count = 0;
    while (a != 0) {
        int digit = a % 10;
        a = a / 10;
        if (digit == b) {
            count++;
            b = -1;    // so that it won't be counted twice
        }
        if (digit == c) {
            count++;
            c = -1;    // so that it won't be counted twice
        }
    }

    if (count == 2) {
        return true;
    } else {
        return false;
    }
}

public static boolean containsBothDigits(int a, int b, int c) {
    boolean foundB = false, foundC = false;
    while (a != 0) {
        foundB = foundB || a % 10 == b;
        foundC = foundC || a % 10 == c;
        a /= 10;
    }
    return foundB && foundC;
}

public static boolean containsBothDigits(int a, int b, int c) {
    return containsDigit(a, b) && containsDigit(a, c);
}
public static boolean containsDigit(int a, int b) {
    while (a != 0) {
        if (a % 10 == b) {
            return true;
        }
        a = a / 10;
    }
    return false;
}
```