Name _____ CIS 201 Final: Chapters 1-8

May 17, 2011

Directions: This is a closed book, closed notes midterm. Place your answers in the space provided. The point value for each question is indicated. There are a total of ?? points on this examination. Put your answers underneath each question. Your total will be weighted to 100. You have 120 minutes for this final.

1. **(9 pts) Procedural Decomposition:** What is the output of the following program?

```
public class One {
    public static void main (String [] args) {
       method2();
       method1();
       method3();
       method2();
    }

    public static void method1() {
       System.out.println("METHOD 1");
    }

    public static void method2() {
       method1();
       System.out.println("METHOD 2");
    }

    public static void method3() {
       method2();
       System.out.println("METHOD 3");
       method1();
    }
}
```

2. **(9 pts) Definite Loops, Nested Loops:** Write a segment of code, not an entire program, to produce the following output:

```
++++*
+++*-
++*--
+*---
*----
```

3. **(9 pts) Expressions:** For each expression in the left-hand column, indicate its value in the right-hand column. Be sure to list a constant of appropriate type (e.g., `7.0` rather than `7` for a `double`, `String`s in `"quotes"`).

Expression	**Value**
`16 / 3 + 3.2 * 2`	_____
`8 / 7 + "5 / 4" + 8 / 3`	_____
`88 % 10 % 3 * 16 / 10`	_____
`29 / 3 / 2 / 4.0 + 3.6 * 2`	_____
`1.4 + (3 + 2 * 6) / (8 – 14 / 3) * 2.2`	_____

4. **(9 pts) While Loops:** For each call below to the following method, write the output that is produced or the value that is returned:

```
public static void mystery(int a, int b) {
    while (b != 0) {
        if (a > b) {
            System.out.print(a + " ");
            a = a - b;
        } else {
            System.out.print(b + " ");
            b = b - a;
        }
    }
    System.out.println(a);
}
```

Method Call	Output or Value Returned
`mystery(42, 0);`	_____
`mystery(6, 12);`	_____
`mystery(18, 27);`	_____
`mystery(24, 60);`	_____
`mystery(50, 15);`	_____

5. **(9 pts) Booleans** Write a method named `enoughTimeForLunch()` that accepts four integers *hour1*, *minute1*, *hour2*, and *minute2* as parameters. Each pair of parameters represents a time on the 24-hour clock (for example, 1:36 PM would be represented as 13 and 36). The method should return `true` if the gap between the two times is long enough to eat lunch: that is, if the second time is at least 45 minutes after the first time. Otherwise the method should return false.

   You may assume that all parameter values are valid: the hours are both between 0 and 23, and the minute parameters are between 0 and 59. You may also assume that both times represent times in the same day, e.g. the first time won't represent a time today while the second time represents a time tomorrow or the day before. Note that the second time might be earlier than the first time; in such a case, your method should return `false.`

   Here are some example calls to your method and their expected return results:

```
enoughTimeForLunch(11, 00, 11, 59) → true
enoughTimeForLunch(12, 30, 13, 00) → false
enoughTimeForLunch(12, 30, 13, 15) → true
enoughTimeForLunch(14, 20, 17, 02) → true
enoughTimeForLunch(12, 30, 9, 30)  → false
enoughTimeForLunch(12, 00, 11, 55) → false
```

6. **(9 pts) File and String Processing:** Write a static method named `reverseLines()` that accepts a `Scanner` containing an input file as a parameter and that echoes the input file to `System.out` with each line of text reversed. For example, given the following input file:

```
If this method works properly,
the lines of text in this file
will be reversed.

Remember that some lines might be blank.
```

Your method should print the following output:

```
,ylreporp skrow dohtem siht fI
elif siht ni txet fo senil eht
.desrever eb lliw


.knalb eb thgim senil emos taht rebmemeR
```

Notice that some of the input lines can be blank lines.

7. (9 pts) Arrays: Consider the following method:

```
public static int mystery(int[] array) {
    int n = 0;
    for (int i = 0; i < array.length - 1; i++) {
        if (array[i] > array[i + 1]) {
            n++;
        }
    }
    return n;
}
```

In the left-hand column below are specific arrays of integers. Indicate in the right-hand column what value would be returned by method `mystery()` if the integer array in the left-hand column is passed as its parameter.

Array	Value Returned
`int[] a1 = {8};` `mystery(a1);`	_____
`int[] a2 = {14, 7};` `mystery(a2);`	_____
`int[] a3 = {7, 1, 3, 2, 0, 4};` `mystery(a3);`	_____
`int[] a4 = {10, 8, 9, 5, 6};` `mystery(a4);`	_____
`int[] a5 = {8, 10, 8, 6, 4, 2};` `mystery(a5);`	_____

8. (9 pts) Arrays

Write a static method named `minGap()` that accepts an integer array as a parameter and returns the minimum 'gap' between adjacent values in the array. The gap between two adjacent values in a array is defined as the second value minus the first value. For example, suppose a variable called array is an array of integers that stores the following sequence of values.

```
int[] array = {1, 3, 6, 7, 12};
```

The first gap is 2 (3 - 1), the second gap is 3 (6 - 3), the third gap is 1 (7 - 6) and the fourth gap is 5 (12 - 7). Thus, the call of `minGap(array)` should return 1 because that is the smallest gap in the array. Notice that the minimum gap could be a negative number. For example, if array stores the following sequence of values:

```
int [] array = {3, 5, 11, 4, 8};
```

The gaps would be computed as 2 (5 - 3), 6 (11 - 5), -7 (4 - 11), and 4 (8 - 4). Of these values, -7 is the smallest, so it would be returned. You may assume that the array has at least length 2.

9. (9 pts) Classes Consider the following class called `Date` class. Each `Date` object represents a calendar date such as September 19th.

```
public class Date {
    private int month;
    private int day;

    public Date(int m, int d) {
        month = m;
        day = d;
    }

    public int getDay() {
        return day;
    }

    public int getMonth() {
        return month;
    }

    // your method would go here
}
```

Write a method named `numDays()` to be placed inside the `Date` class (although, please write this method on the next page). This method returns the number of days in a `Date` object's month. For example:

```
Date d = new Date(12, 7);
System.out.println(d.numDays());
```

would result in 31 being printed out because there are 31 days in month 12 (December). January, March, May, July, August, October, and December all have 31 days. The other months have 30 days except for February which has 28. You can ignore leap year.

10. (9 pts) Classes: Consider the `Date` class in Question 9 once more. The method below is to be added to the `Date` class, and it is supposed to set a `Date` object to a new `Date`. For example after the following code executes

```
Date d = new Date (12, 4);
d.setDate(3,10);
```

`Date d` should be now set to March 10. Does it work correctly? If it does, answer yes. If it does not, correct it.

```
public void setDate(int m, int d) {
    int month = m;
    int day = d;
}
```