CSE 142 SPRING 2007 MIDTERM

1. Expressions (10 points)

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"). If the expression is illegal, then write "error".

Expression	Value
`0.5 + 3 / 2.0 * 1 - 5 / 10`	
`5 + 11 % 6 + 34 % 4 % 8`	
`2 * 10 + 3 / 2 / 1.0`	
`6 - 5 + 4 * 3 - 2 + 1`	
`8 + 2 * -10 + 102 + "2" + 5 * 2`	

2. Parameter Mystery (20 points)

At the bottom of the page, write the output produced by the following program, as it would appear on the console.

```
public class ParameterMystery {
    public static void main(String[] args) {
        String a = "apple";
        String p = "lead";
        String q = "king";
        String r = "people";
        String king = "run";
        String and = "bear";
        String chicken = "pork";

        mystery(p, r, q);
        mystery(q, p, king);
        mystery(p, king, "chicken");
        mystery(q, and, "a");
        mystery(r, "and", r);
    }

    public static void mystery(String r, String p, String q) {
        System.out.println(p + " see " + q + " " + r);
    }
}
```

3. While Loop Simulation (15 points)

For each call below to the following method, write the output that is printed, as it would appear on the console:

```
public static void mystery(int a, int b) {
    while (a > 0) {
        a = a - b;
        b++;

        System.out.print(a + " ");
    }
    System.out.println(b);
}
```

Method Call | Output

```
mystery(7, 8);

mystery(11, 7);

mystery(25, 10);

mystery(-6, 10);

mystery(10, 3);
```

4. Assertions (15 points)

For each of the five points labeled by comments, identify each of the following assertions as being either always true, never true or sometimes true / sometimes false.

```
public static int mystery(int a) {
    int b = 1;

    // Point A
    while (a > 0) {
        // Point B
        b += a;

        if (b > a) {
            // Point C
            b = -b;
        }

        // Point D
        a--;
    }

    // Point E
    return b;
}
```

	a < b	a > 0	b > 0
Point A			
Point B			
Point C			
Point D			
Point E			

5. Conditionals (6 points)

Consider the following method:

```
public static void mystery(int x) {
    if (x != 3) {
        System.out.println("Alpha");
    } else if (x > 8) {
        System.out.println("Bravo");
    } else {
        System.out.println("Charlie");
    }
    if (x < 2) {
        System.out.println("Delta");
    }
}
```

(a) Under what conditions will the method print out only "`Bravo`"? Explain your answer using 20 words or less.

(b) Under what conditions will the method print out nothing? Explain your answer using 20 words or less.

6. Program Mystery (3 points)

What does the following program print?

```
public class Mystery {
    public static void main(String[] args) {
        int sum = 0;
        computeSum(4, sum);
        System.out.println("sum = " + sum);
    }

    public static void computeSum(int max, int sum) {
        for (int i = 1; i <= max; i++) {
            sum += i;
        }
    }
}
```

7. Programming (15 points)

Write a static method named `areReversals` that accepts two strings and returns true if the strings are reversals of each other, i.e. if one word is equal to the other word spelled backwards. Case does not matter.

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

Call	Value Returned
`areReversals("hello", "goodbye")`	`false`
`areReversals("hello", "olleh")`	`true`
`areReversals("HELLO", "olleh")`	`true`
`areReversals("hello", "aolleh")`	`false`

8. Programming (15 points)

In the number-guessing game, you are to guess a number between 1 and 100. After each guess, you are given a hint as to whether your guess is higher or lower than the actual number. Your goal is to guess the actual number with as few guesses as possible. Consider how you would play such a game.

The optimal strategy would make an initial guess of 50, because it is halfway between 1 and 100. The first hint would indicate whether 50 was higher or lower than the actual number. Using that information, the optimal strategy can narrow the interval of numbers to consider:

- If 50 is higher than the actual number, then the actual number is in the interval between 1 and 49. The optimal player would then guess 25, which is halfway between 1 and 49.
- If 50 is lower than the actual number, then the actual number is in the interval between 51 and 100. The optimal player would then guess 75, which is halfway between 51 and 100.

After each guess, the optimal strategy continually halves the size of the interval in which to make guesses until the answer is reached.

Write a static method named `printGuesses` that accepts the actual number (an integer) and prints the series of guesses that would be made by an optimal player. The guesses are separated by commas.

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

Call	Output
`printGuesses(34);`	`50, 25, 37, 31, 34`
`printGuesses(1);`	`50, 25, 12, 6, 3, 1`
`printGuesses(82);`	`50, 75, 88, 81, 84, 82`
`printGuesses(50);`	`50`
`printGuesses(100);`	`50, 75, 88, 94, 97, 99, 100`