

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

<u>Expression</u>	<u>Value</u>
6 - 3 * 2 + 1 + "6"	"16"
6 % 4 % 9 + 5 - 13 % 5	4
2 + 4 * 2 / 3 * 3 + 1	9
1 * 2 + 3 * 4 + 5 + 6	25
0.25 + 3 / 4 / 3 + 0.75	1.0

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 green = "i";
        String am = "green";
        String ham = "sam";
        String i = "eggs";
        String eggs = "am";
        String sam = "ham";

        mystery(sam, i, am);
        mystery(ham, green, eggs);
        mystery(green, eggs, ham);
        mystery(i, sam, "green");
        mystery(i, "am", sam);
    }

    public static void mystery(String eggs, String ham, String green) {
        System.out.println("I do not like " + green + " " + eggs + " and " + ham);
    }
}
```

```
I do not like green ham and eggs
I do not like am sam and i
I do not like sam i and am
I do not like green eggs and ham
I do not like ham eggs and am
```

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 (b > 0) {
        a = a - b;
        b = a - b;

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

<u>Method Call</u>	<u>Output</u>
mystery(16, 4);	12 8 4 -4
mystery(11, -7);	
mystery(25, 10);	15 5 10 5 5 0
mystery(-6, 10);	-16 -26
mystery(10, 3);	7 4 3 -1

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 void mystery(int a) {
    int b = 0;

    // Point A
    while (a > 0) {
        // Point B
        if (a % 2 == 0) {
            b++;
            // Point C
        } else {
            a--;
        }

        // Point D
        a /= 2;
    }

    // Point E
}
```

	a % 2 == 0	a > 0	b > 0
Point A	SOMETIMES	SOMETIMES	NEVER
Point B	SOMETIMES	ALWAYS	SOMETIMES
Point C	ALWAYS	ALWAYS	ALWAYS
Point D	ALWAYS	SOMETIMES	SOMETIMES
Point E	SOMETIMES	NEVER	SOMETIMES

5. Conditionals (9 points)

Consider the following piece of code where `a` and `b` are Boolean variables:

```
if (a && b) {
    System.out.print("Hi");
} else if (b) {
    System.out.println(" there!");
}
```

(a) Under what conditions will the code print out only " there! "? Explain your answer using 20 words or less.

If `a` is `false` and `b` is `true`, then the first condition will fail, but the second will not.

(b) Under what conditions will the code print out "Hi there! "? Explain your answer using 20 words or less.

Never. Both branches cannot execute. Either exactly one branch executes or none do.

6. Programming (15 points)

Morse code is a translation from letters to a series of dots and dashes. For example, the first five letters are translated as follows:

A = .- B = -... C = -.-. D = -.. E = .

Write a static method named `printInMorse` that accepts a word as a parameter and prints out the Morse version of the word. Assume words only consist of the above letters. Case does not matter. Letters in Morse code are separated by spaces (it is OK to have a space after the last Morse letter). If it helps, you can write helper methods (extra methods that get used by the primary method).

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

Call	Output
<code>printInMorse("bead")</code>	<code>-... .- -..</code>
<code>printInMorse("Dad")</code>	<code>-.. .- -..</code>
<code>printInMorse("aBcDe")</code>	<code>.- -... -.-. -.. .</code>
<code>printInMorse("cab")</code>	<code>-.-. .- -...</code>

One possible solution:

```
public static void printInMorse(String word) {
    word = word.toLowerCase();

    for (int i = 0; i < word.length(); i++) {
        char c = word.charAt(i);
        if (c == 'a') {
            System.out.print(".- ");
        } else if (c == 'b') {
            System.out.print("-... ");
        } else if (c == 'c') {
            System.out.print("-.-. ");
        } else if (c == 'd') {
            System.out.print("-.. ");
        } else { // c == 'e'
            System.out.print(". ");
        }
    }
}
```

7. Programming (15 points)

Write a static method named `battleToTheDeath` that simulates a fight between two players. It accepts two integer parameters representing the minimum and maximum amount of damage a player can inflict upon the other player. The "health" of a player is represented by a number initially set to 100. Each player randomly attacks the other, subtracting damage from the attacked player's health, until one of the player's health falls below 1. The damage is a random number between the minimum damage value and the maximum damage value (inclusive).

You are to reproduce the format of the sample output below. Here is a sample run for the call `battleToTheDeath(20,40)`:

Call	Output
<code>battleToTheDeath(20,40);</code>	DEATH MATCH!!! Player 2 attacks! -40 damage. Player 1 attacks! -31 damage. Player 1 attacks! -32 damage. Player 2 attacks! -23 damage. Player 1 attacks! -35 damage. Player 2 attacks! -30 damage. Player 1 attacks! -28 damage. Player 1 wins!

You may assume that the parameters passed to your method will be non-negative and the maximum damage value (the second parameter) will always be greater than the minimum damage value (the first parameter).

```
public static void battleToTheDeath(int minDamage, int maxDamage) {
    System.out.println("DEATH MATCH!!!");
    Random r = new Random();
    int player = 0;

    int playerOneHealth = 100, playerTwoHealth = 100;
    while (playerOneHealth > 0 && playerTwoHealth > 0) {
        int damage = r.nextInt(maxDamage - minDamage + 1) + minDamage;
        player = r.nextInt(2) + 1;

        if (player == 1) {
            playerTwoHealth -= damage;
        } else {
            playerOneHealth -= damage;
        }

        System.out.println("Player " + player + " attacks! -" + damage + " damage.");
    }
    System.out.println("Player " + player + " wins!");
}
```