Sample Midterm Exam #2

1. 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
`8 + 5 * 3 / 2`	_____
`1.5 * 4 * 7 / 8 + 3.4`	_____
`73 % 10 - 6 % 10 + 28 % 3`	_____
`4 + 1 + 9 + "." + (-3 + 10) + 11 / 3`	_____
`3 / 14 / 7 / (1.0 * 2) + 10 / 6`	_____
`10 > 11 == 4 / 3 > 1`	_____

2. Parameter Mystery

At the bottom of the page, write the output produced by the following program.

```
public class ParameterMystery {
    public static void main(String[] args) {
        String x = "happy";
        String y = "pumpkin";
        String z = "orange";
        String pumpkin = "sleepy";
        String orange = "vampire";

        orange(y, x, z);
        orange(x, z, y);
        orange(pumpkin, z, "y");
        z = "green";
        orange("x", "pumpkin", z);
        orange(y, z, orange);
    }

    public static void orange(String z, String y, String x) {
        System.out.println(y + " and " + z + " were " + x);
    }
}
```

3. If/Else Simulation

For each call of the method below, write the value that is returned:

```
public static int mystery(int n) {
    if (n < 0) {
        n = n * 3;
        return n;
    } else {
        n = n + 3;
    }
    if (n % 2 == 1) {
        n = n + n % 10;
    }
    return n;
}
```

Method Call	Value Returned
mystery(-5)	_____
mystery(0)	_____
mystery(7)	_____
mystery(18)	_____
mystery(49)	_____

4. While Loop Simulation

For each call of the method below, write the value that is returned:

```
public static int mystery(int i, int j) {
    int k = 0;
    while (i > j) {
        i = i - j;
        k = k + (i - 1);
    }
    return k;
}
```

Method Call	Value Returned
mystery(2, 9)	_____
mystery(5, 1)	_____
mystery(38, 5)	_____
mystery(5, 5)	_____
mystery(40, 10)	_____

5. Assertions

For the following method, identify each of the three assertions in the table below as being either ALWAYS true, NEVER true or SOMETIMES true / sometimes false at each labeled point in the code. You may abbreviate these choices as A/N/S respectively.

```
public static int mystery(Scanner console) {
    int y = 0;
    int z = 1;
    int next = console.nextInt();

    // Point A
    while (next >= 0) {
        // Point B
        if (y > z) {
            // Point C
            z = y;
        }
        y++;
        next = console.nextInt();
        // Point D
    }

    // Point E
    return z;
}
```

	next < 0	y > z	y == 0
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming

Write a static method named `printMultiples` that accepts an integer n and an integer m as parameters and that prints a complete line of output reporting the first m multiples of n. For example, the following calls:

```
printMultiples(3, 5);
printMultiples(7, 3);
```

should produce this output:

```
The first 5 multiples of 3 are 3, 6, 9, 12, 15
The first 3 multiples of 7 are 7, 14, 21
```

Notice that the multiples are separated by commas. You must exactly reproduce this format. You may assume that the number of multiples you will be asked to generate is greater than or equal to 1.

7. Programming

Write a static method named `monthApart` that accepts four integer parameters representing two calendar dates. Each date consists of a month (1 through 12) and a day (1 through the number of days in that month [28-31]). Assume that all dates occur during the same year. The method returns whether the dates are at least a month apart. For example, the following dates are all considered to be at least a month apart from 9/19 (September 19): 2/14, 7/25, 8/2, 8/19, 10/19, 10/20, and 11/5. The following dates are NOT at least a month apart from 9/19: 9/20, 9/28, 10/1, 10/15, and 10/18. Note that the first date could come before or after (or be the same as) the second date. Assume that all parameter values passed are valid.

Sample calls:

```
monthApart( 6, 14,  9, 21) should return true,  because June 14 is at least a month before September 21
monthApart( 4,  5,  5, 15) should return true,  because April 5 is at least a month before May 15
monthApart( 4, 15,  5, 15) should return true,  because April 15 is at least a month before May 15
monthApart( 4, 16,  5, 15) should return false, because April 16 isn't at least a month apart from May 15
monthApart( 6, 14,  6,  8) should return false, because June 14 isn't at least a month apart from June 8
monthApart( 7,  7,  6,  8) should return false, because July 7 isn't at least a month apart from June 8
monthApart( 7,  8,  6,  8) should return true,  because July 8 is at least a month after June 8
monthApart(10, 14,  7, 15) should return true,  because October 14 is at least a month after July 15
```

8. Programming

Write a static method named `threeHeads` that repeatedly flips a coin until three heads *in a row* are seen. You should use a `Random` object to give an equal chance to a head or a tail appearing. Each time the coin is flipped, what is seen is displayed (H for heads, T for tails). When 3 heads in a row are flipped a congratulatory message is printed. Here are possible outputs of two calls to `threeHeads`:

```
T T T H T H H H
Three heads in a row!
T H T H T T T T T H H T H H H
Three heads in a row!
```

Sample Midterm Exam #2 Key

Also check out Practice-It to test solving these problems or to type in your own solution to see if it works!

1. Expressions

Expression	Value
`8 + 5 * 3 / 2`	`15`
`1.5 * 4 * 7 / 8 + 3.4`	`8.65`
`73 % 10 - 6 % 10 + 28 % 3`	`-2`
`4 + 1 + 9 + "." + (-3 + 10) + 11 / 3`	`"14.73"`
`3 / 14 / 7 / (1.0 * 2) + 10 / 6`	`1.0`
`10 > 11 == 4 / 3 > 1`	`true`

2. Parameter Mystery

```
happy and pumpkin were orange
orange and happy were pumpkin
orange and sleepy were y
pumpkin and x were green
green and pumpkin were vampire
```

3. If/Else Simulation

Method Call	Value Returned
`mystery(-5)`	`-15`
`mystery(0)`	`6`
`mystery(7)`	`10`
`mystery(18)`	`22`
`mystery(49)`	`52`

4. While Loop Simulation

Method Call	Value Returned
`mystery(2, 9)`	`0`
`mystery(5, 1)`	`6`
`mystery(38, 5)`	`119`
`mystery(5, 5)`	`0`
`mystery(40, 10)`	`57`

5. Assertions

	`next < 0`	`y > z`	`y == 0`
Point A	SOMETIMES	NEVER	ALWAYS
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	NEVER	ALWAYS	NEVER
Point D	SOMETIMES	SOMETIMES	NEVER
Point E	ALWAYS	SOMETIMES	SOMETIMES

6. Programming (one solution shown)

```
public static void printMultiples(int n, int times) {
    System.out.print("The first " + times + " multiples of " + n + " are " + n);
    for (int i = 2; i <= times; i++) {
        System.out.print(", " + i * n);
    }
    System.out.println();
}
```

7. Programming (four solutions shown)

```
public static boolean monthApart(int m1, int d1, int m2, int d2) {
    if (m1 == m2) {
        return false;
    } else if (m1 <= m2 - 2) {
        return true;
    } else if (m1 >= m2 + 2) {
        return true;
    } else if (m1 == m2 - 1) {
        if (d1 <= d2) {
            return true;
        } else {
            return false;
        }
    } else if (m1 == m2 + 1) {
        if (d1 >= d2) {
            return true;
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

```
public static boolean monthApart(int m1, int d1, int m2, int d2) {
    if (m1 < m2 - 1 || m1 > m2 + 1) {
        return true;
    } else if (m1 == m2 - 1 && d1 <= d2) {
        return true;
    } else if (m1 == m2 + 1 && d1 >= d2) {
        return true;
    } else {
        return false;
    }
}
```

```
public static boolean monthApart(int m1, int d1, int m2, int d2) {
    return (m2 - m1 > 1) || (m1 - m2 > 1) ||
           (m2 - m1 == 1 && d1 <= d2) ||
           (m1 - m2 == 1 && d1 >= d2);
}
```

```
public static boolean monthApart(int m1, int d1, int m2, int d2) {
    return Math.abs((m1 * 31 + d1) - (m2 * 31 + d2)) >= 31;
}
```

8. Programming (one solution shown)

```
public static void threeHeads() {
    Random r = new Random();
    int numHeads = 0;

    while (numHeads < 3) {
        int flip = r.nextInt(2); // flip coin
        if (flip == 0) { // heads
            numHeads++;
            System.out.print("H ");
        } else {
            numHeads = 0;
            System.out.print("T ");
        }
    }
    System.out.println();
    System.out.println("Three heads in a row!");
}
```

Building Java Programs
Sample Final Exam #2

Name of Student_____

Section (e.g., AA)_____ TA_____

Exam time (10:30, 12:30)_____

This is an open-book/open-note exam. Space is provided for your answers. Use the backs of pages if necessary. The exam is divided into ten questions with the following points:

#	Problem Area	Points	Score
1	Expressions	5	_____
2	Simulation	10	_____
3	Polymorphism	6	_____
4	Token-Based File processing	9	_____
5	Line-Based File Processing	10	_____
6	Arrays	10	_____
7	ArrayList	10	_____
8	Critters	15	_____
9	Arrays	15	_____
10	Programming	10	_____
	Total	100	_____

In general the exam is not graded on style and you do not need to include comments. You do not have to include any import statements.

You may use a calculator to do simple arithmetic, although you are not allowed to program the calculator or to run a stored program.

Do not begin work on this exam until instructed to do so. Any student who starts early or who continues to work after time is called will receive a 10 point penalty.

If you finish the exam early, please hand your exam to the instructor and exit quietly through the front door.

1. Expressions, 5 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).

Expression	Value
`(6 + 7)/4/2.0`	_____
`13/2 * 3 % 10 - 1`	_____
`30/(2 * 6) + 1.5`	_____
`13 % 5 + 5 * 3/4`	_____
`"3 * 4" + 3 * 4 + 10`	_____

2. Simulation, 10 points. You are to simulate the execution of a method that manipulates an array of integers. Consider the following method:

```
public static void mystery(int[] list) {
    for (int i = 2; i < list.length; i++) {
        list[i] = list[i] + list[i - 1] + list[i - 2];
    }
}
```

In the left-hand column below are specific lists of integers. You are to indicate in the right-hand column what values would be stored in the list after method mystery executes if the integer list in the left-hand column is passed as a parameter to mystery.

Original List	Final List
(8)	_____
()	_____
(3, 0, 1, 4, 7)	_____
(0, 1, 2, 3, 4, 5)	_____
(7, 4, -10, 8, 2)	_____

3. Polymorphism, 6 points. Assume the following classes have been defined:

```
public class Mammal extends SeaCreature {
    public void method1() {
        System.out.println("warm-blooded");
    }
}

public class SeaCreature {
    public void method1() {
        System.out.println("creature 1");
    }

    public void method2() {
        System.out.println("creature 2");
    }

    public String toString() {
        return "ocean-dwelling";
    }
}

public class Whale extends Mammal {
    public void method1() {
        System.out.println("spout");
    }

    public String toString() {
        return "BIG!";
    }
}

public class Squid extends SeaCreature {
    public void method2() {
        System.out.println("tentacles");
    }

    public String toString() {
        return "squid";
    }
}
```

Consider the following code fragment:

```
SeaCreature[] elements = {new Squid(), new Whale(), new SeaCreature(),
                          new Mammal()};
for (int i = 0; i < elements.length; i++) {
    System.out.println(elements[i]);
    elements[i].method1();
    elements[i].method2();
    System.out.println();
}
```

What output is produced by this code?

4. Token-Based File Processing, 9 points. Write a static method processData that takes as a parameter a Scanner holding a sequence of integers and that reports each of the cumulative sums of the sequence along with the average of the numbers. For example, if the Scanner contains the following data:

```
8 4 2 9 7 13 5 9
```

your method should produce the following output:

```
Sum of 1 = 8
Sum of 2 = 12
Sum of 3 = 14
Sum of 4 = 23
Sum of 5 = 30
Sum of 6 = 43
Sum of 7 = 48
Sum of 8 = 57
Average = 7.125
```

Notice that the various lines of output report the sum including just the first number, then including the first two numbers, then including the first three numbers, and so on, up to the sum including all numbers. The final line of output reports the average of the sequence of numbers. Notice that this is the average of the numbers themselves, not the average of the cumulative sums.

The amount of output will vary depending upon how many numbers are in the sequence. For example, if the Scanner contains the following values:

```
1 2 3 4
```

the method should produce the following output:

```
Sum of 1 = 1
Sum of 2 = 3
Sum of 3 = 6
Sum of 4 = 10
Average = 2.5
```

You are to exactly reproduce the format of these sample outputs. You may assume that the Scanner has at least one integer to be processed.

Write your solution to processData below.

5. Line-Based File Processing, 10 points. Write a static method processFile that takes a Scanner containing an input file as a parameter and that prints the output file with certain lines underlined. The lines to be underlined will all begin with a period. The period should not be printed. You should print the text that follows the period on a line by itself followed by a line of dashes that is equal in length to the text that follows the period.

   For example, given the following input file:

```
.Statement of Purpose
I didn't expect to major in computer science until I took cse142.
I liked it more than I expected and that got me hooked on cs.

.High School Performance
I got very good grades in high school, graduating in the top 10% of
my class.

.College Performance
I have done well in my college classes, with an overall gpa of 3.5.
```

   Your method should print the following output:

```
Statement of Purpose
--------------------
I didn't expect to major in computer science until I took cse142.
I liked it more than I expected and that got me hooked on cs.

High School Performance
-----------------------
I got very good grades in high school, graduating in the top 10% of
my class.

College Performance
-------------------
I have done well in my college classes, with an overall gpa of 3.5.
```

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

   Write your solution to processFile below.

6. Arrays, 10 points. Write a static method printNumber that takes an array of integers that represent the digits of a large integer, as in assignment #6, and that prints the integer with commas between every 3 digits starting from the right. You may assume that the values in the array are all between 0 and 9 and that there are no leading 0's. For example, if the array contains the digits (1, 5, 0, 0), then your method should produce the following output:

```
1,500
```

If the array instead contains the digits (3, 8, 4, 9, 2, 1, 4, 7), then your method should produce the following output:

```
38,492,147
```

Your method might not print any commas if the number is small enough. For example, if the array contains the digits (2, 5, 0), then your method should produce the following output:

```
250
```

Your method should produce an entire line of output so that if it is called several times in a row, each call will produce a separate line of output.

Write your solution to printNumber below.

7. ArrayList, 10 points. Write a static method manyStrings that takes an ArrayList of Strings and an integer n as parameters and that replaces every String in the original list with n of that String. For example, suppose that an ArrayList called "list" contains the following values:

("squid", "octopus")

And you make the following call:

manyStrings(list, 2);

Then list should store the following values after the call:

("squid", "squid", "octopus", "octopus")

As another example, suppose that list contains the following:

("a", "a", "b", "c")

and you make the following call:

manyStrings(list, 3);

Then list should store the following values after the call:

("a", "a", "a", "a", "a", "a", "b", "b", "b", "c", "c", "c")

You may assume that the ArrayList you are passed contains only Strings and that the integer n is greater than 0.

Recall that the primary methods for manipulating an ArrayList are:

add(Object value)	appends value at end of list
add(int index, Object value)	inserts given value at given index, shifting subsequent values right
get(int index)	returns the value at given index
remove(int index)	removes value at given index, shifting subsequent values left
set(int index, Object value)	replaces value at given index with given value
size()	returns the number of elements in list

Write your solution to manyStrings below.

8. Critters, 15 points. Write a class Ant that implements the Critter interface. The instances of the Ant class go through cycles of ten moves. For each cycle, the Ant randomly chooses between going north-east or going south-west (each choice should be equally likely). For a north-east cycle, the ant goes north 5 times and then goes east 5 times. For a south-west cycle, the ant goes south 5 times and then goes west 5 times. Each Ant object should return the letter "A" for display purposes.

   Remember that the Critter interface is defined as follows:

```
public interface Critter {
    public char getChar();

    public int getMove();

    <definitions for constants NORTH, SOUTH, EAST and WEST>
}
```

   Write your solution to the Ant class below.

9. Arrays, 15 points. Write a static method interleave that takes two arrays of integers as parameters and that returns a new array that contains the result of interleaving the elements of the two arrays. It should not alter either of its parameters. Two arrays are interleaved by taking elements in the following order:

```
1st element of 1st list
1st element of 2nd list
2nd element of 1st list
2nd element of 2nd list
3rd element of 1st list
3rd element of 2nd list
and so on
```

For example, if the variables list1 and list2 contain the following values:

```
list1: (1, 8, 3, 9)
list2: (2, 12, 6, 14)
```

Then the call interleave(list1, list2) should return a new array that stores the following values:

```
(1, 2, 8, 12, 3, 6, 9, 14)
```

The order of the parameters matters. For example, interleave(list2, list1) should produce the following array as its result:

```
(2, 1, 12, 8, 6, 3, 14, 9)
```

One list might be longer than the other, in which case any extra values from the longer list should simply be appended at the end of the result. For example, given the following lists:

```
list1: (1, 8, 3, 9)
list2: (82, 7, 4, 2, 1, 6, 5, 0, 18)
```

The call interleave(list1, list2) should produce the following result:

```
(1, 82, 8, 7, 3, 4, 9, 2, 1, 6, 5, 0, 18)
```

Notice that the first four values of the two arrays have been interleaved and the excess values from the second list (1, 6, 5, 0, 18) have been included at the end of the result. In this case the second list was longer, but it could be the case that the first list is longer. Either list could also be empty, in which case the result should contain the values from the other list. Notice that your method is to return a new array.

Write your solution to interleave below.

10. Programming, 10 points. Write a static method indexOf that takes two arrays of integers and that returns the index of the first occurrence of the first list in the second list or -1 if the first list does not appear in the second list. For example, suppose that you have two integer arrays called list1 and list2 that store the following values:

    list1: (1, 3, 5, 8, 12, 1, 3, 17, 1, 3, 6, 9, 1, 3, 6)
    list2: (1, 3, 6)

    then the call:

    indexOf(list2, list1)

    should return 8 because the sequence of values stored in list2 appears in list1 starting with index 8. Notice that list2 actually appears twice in list1, starting at position 8 and starting at position 12. Your method is to return the first such position.

    If the second list is not contained in the first list, then the method should return the value -1. For example, if list1 had the same value as before but list2 stored (12, 1, 3, 6), then the call indexOf(list2, list1) should return -1 because list2 is not contained in list1. If the first list is empty, your method should return 0.

    Write your solution to indexOf below.

Building Java Programs
Sample Final Exam #2 Solution Key

1.

Expression	Value
(6 + 7)/4/2.0	1.5
13/2 * 3 % 10 - 1	7
30/(2 * 6) + 1.5	3.5
13 % 5 + 5 * 3/4	6
"3 * 4" + 3 * 4 + 10	"3 * 41210"

2.

Original List	Final List
(8)	(8)
()	()
(3, 0, 1, 4, 7)	(3, 0, 4, 8, 19)
(0, 1, 2, 3, 4, 5)	(0, 1, 3, 7, 14, 26)
(7, 4, -10, 8, 2)	(7, 4, 1, 13, 16)

3. Polymorphism. The output produced is as follows.

```
squid
creature 1
tentacles

BIG!
spout
creature 2

ocean-dwelling
creature 1
creature 2

ocean-dwelling
warm-blooded
creature 2
```

4. Token-Based File Processing. One possible solution appears below.

```
public static void processData(Scanner input) {
    int count = 0;
    int sum = 0;
    while (input.hasNextInt()) {
        sum += input.nextInt();
        count++;
        System.out.println("Sum of " + count + " = " + sum);
    }
    double average = (double)sum/count;
    System.out.println("Average = " + average);
}
```

5. Line-Based File Processing. One possible solution appears below.

```
public static void processFile(Scanner input) {
    while (input.hasNextLine()) {
        String text = input.nextLine();
        if (text.length() == 0 || text.charAt(0) != '.')
            System.out.println(text);
        else {
            System.out.println(text.substring(1));
            for (int i = 1; i <= text.length() - 1; i++)
                System.out.print("-");
            System.out.println();
        }
    }
}
```

6. Arrays. One possible solution appears below.

```
public static void printNumber(int[] digits) {
    for (int i = 0; i < digits.length; i++) {
        if (i > 0 && (digits.length - i) % 3 == 0)
            System.out.print(",");
        System.out.print(digits[i]);
    }
    System.out.println();
}
```

7. ArrayLists. One possible solution appears below.

```
public static void manyStrings(ArrayList list, int n) {
    for (int i = list.size() - 1; i >= 0; i--) {
        String target = (String)list.get(i);
        for (int j = 0; j < n - 1; j++)
            list.add(i, target);
    }
}
```

8. Critters. One possible solution appears below.

```
public class Ant implements Critter {
    private int move;
    private int direction1;
    private int direction2;

    public char getChar() {
        return 'A';
    }

    public int getMove() {
        if (this.move % 10 == 0)
            if (Math.random() < 0.5) {
                this.direction1 = NORTH;
                this.direction2 = EAST;
            } else {
                this.direction1 = SOUTH;
                this.direction2 = WEST;
            }
        int result;
        if (this.move % 10 < 5)
            result = this.direction1;
        else
            result = this.direction2;
        this.move++;
        return result;
    }
}
```

9. Arrays. One possible solution appears below.

```
public static int[] interleave(int[] list1, int[] list2) {
    int[] result = new int[list1.length + list2.length];
    int min = Math.min(list1.length, list2.length);
    for (int i = 0; i < min; i++) {
        result[2 * i] = list1[i];
        result[2 * i + 1] = list2[i];
    }
    for (int i = min; i < list1.length; i++)
        result[i + min] = list1[i];
    for (int i = min; i < list2.length; i++)
        result[i + min] = list2[i];
    return result;
}
```

10. Programming. One possible solution appears below.

```
public static int indexOf(int[] target, int[] list) {
    for (int i = 0; i < list.length - target.length; i++) {
        boolean ok = true;
        for (int j = 0; j < target.length; j++)
            if (list[i + j] != target[j])
                ok = false;
        if (ok)
            return i;
    }
    return -1;
}
```