CSE 142
Sample Final Exam #4

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
`!(4 > 3.5) || ((6 < 3) && (4 < 9))`	_____
`99 % 100 + 10 / 4 / 1.0`	_____
`"8" + 3 * 2 + (3 * 25) + 3 * 10 + 3 * 3`	_____
`6 / 2 + (14 < 3)`	_____
`-1 + 11 % 7 * 2 + 1000 + 110 * 3`	_____

2. Array Mystery (15 points)

Consider the following method:

```
public static void mystery(int[] array) {
    for (int i = array.length - 2; i >= 0; i--) {
        array[i] = array[i] + array[i + 1];

    }
}
```

Indicate in the right-hand column what values would be stored in the array after the method `mystery` executes if the integer array in the left-hand column is passed as a parameter to `mystery`.

Array	Final Contents of Array
`int[] a1 = {8, 9};` `mystery(a1);`	_____
`int[] a2 = {14, 7, 1};` `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);`	_____

3. Mystery (4 points)
What does the following block of code print?

```
int a = 5;
int b = 2;

if (b > 3 || a < b) {
   System.out.println("Go Huskies!!");
} else if (b > 0 && a < 8) {
   System.out.print("Go ");
} else if (b == 2) {
   System.out.println("Huskies!!");
} else {
   System.out.println("Go Huskies!!");
}
```

4. Mystery (10 points)
What does the following program print?

```
public class Mystery {
    public static void main(String[] args) {
        String fish = "one";
        String two = "fish";
        String one = "red";
        String blue = "two";
        String red = "blue";

        mystery(two, blue, fish);
        mystery(one, two, red);
        fish = "blue";
        mystery("fish", fish, one);
    }

    public static void mystery(String fish, String two, String one) {
        System.out.println(one + " " + fish + ", " + two + " " + "fish");
    }
}
```

5. Inheritance Mystery (10 points)

Assume that the following classes have been defined:

```
public class Brian extends Lois {
    public void b() {
        a();
        System.out.print("Brian b   ");
    }

    public String toString() {
        return "Brian";
    }
}

public class Lois extends Meg {
    public void a() {
        System.out.print("Lois a   ");
        super.a();
    }

    public void b() {
        System.out.print("Lois b   ");
    }
}
```

```
public class Meg {
    public void a() {
        System.out.print("Meg a   ");
    }

    public void b() {
        System.out.print("Meg b   ");
    }

    public String toString() {
        return "Meg";
    }
}

public class Stewie extends Brian {
    public void a() {
        super.a();
        System.out.print("Stewie a   ");
    }

    public String toString() {
        return super.toString() + " Stewie";
    }
}
```

Given the classes above, what output is produced by the following code?

```
Meg[] elements = {new Lois(), new Stewie(), new Meg(), new Brian()};
for (int i = 0; i < elements.length; i++) {
    elements[i].a();
    System.out.println();
    elements[i].b();
    System.out.println();
    System.out.println(elements[i]);
    System.out.println();
}
```

6. Programming (10 points)

Write a static method `isSumArray` that accepts an array of integers and returns whether for every group of three elements in the array, the first two elements sum up to the third. If the size of the array cannot be divided into groups of three, then the array does not pass the test.

For example, given the following arrays:

```
int[] array1 = { 1, 2, 3, 8, 7, 15, 9, 3, 12 };
int[] array2 = { 1, 2, 3, 4, 5 };
int[] array3 = { 6, 11, 2008 };
int[] array4 = { -4, 7, 3, 8, -2, 6 };
```

Calling `isSumArray` will result in the following values:

Call	Value Returned
`isSumArray(array1)`	`true`
`isSumArray(array2)`	`false`
`isSumArray(array3)`	`false`
`isSumArray(array4)`	`true`

In the first array, for every group of three numbers (1-2-3, 8-7-15, and 9-3-12), the first two numbers add up to the third. The second array cannot be divided into groups of three. The third array can be divided, but the first two numbers do not add up to the third.

7. Programming (15 points)

Write a static method `printWinner` that accepts a `Scanner` holding a sequence of names and numbers. Following each name is a series of one or more numbers. A person's sum is the total of the numbers until the next name. The method will print out the name of the person who has the highest sum less than or equal to 21. If everyone's sum is over 21, then the method will print "Everyone busted!" You may assume that there is at least one name.

For example, given the following Scanners:

```
Scanner input1 = new Scanner("alpha 10 5 9 bravo 8 charlie 11 9");
Scanner input2 = new Scanner("delta 10 3 9 echo 3 1 10 10");
Scanner input3 = new Scanner("foxtrot 11 5 6");
Scanner input4 = new Scanner("golf 4 8 hotel 10 6 india 9 8 7");
```

Calling `printWinner` will result in the following output:

Call	Output
`printWinner(input1)`	`charlie is the winner!`
`printWinner(input2)`	`Everyone busted!`
`printWinner(input3)`	`Everyone busted!`
`printWinner(input4)`	`hotel is the winner!`

In the first example, the respective sums for alpha, bravo, and charlie are 24 (10 + 5 + 9), 8 (8), and 20 (11 + 9). The highest sum that is less than or equal to 21 belongs to charlie. In the second example, the sums are 22 (10 + 3 + 9) and 24 (3 + 1 + 10 + 10). Since no sum is less than or equal to 21, everyone busted.

8. Programming (15 points)

Write a class called `Chameleon` that extends the `Critter` class. The instances of the `Chameleon` class cycle through three different ways of displaying themselves. On their first move they should appear as a red R. On their second move they should appear as a white W. On their third move they should appear as a blue B. Then this pattern repeats itself (red R, white W, blue B, red R, white W, blue B, etc). They should always infect on moves when they are red no matter what is in front of them. On moves when they are white or blue, they should hop when they can and turn right when they can't hop.

9. Programming (10 points)

Write a static method named `hasMirrorTwice` that accepts two arrays of integers *a1* and *a2* as parameters and returns `true` if *a1* contains all the elements of *a2* in reverse order at least twice (and `false` otherwise). For example, if *a2* stores the elements `{1, 2, 3}` and *a1* stores the elements `{6, 3, 2, 1, 4, 1, 3, 2, 1, 5}`, your method would return `true`.

Assume that both arrays passed to your method will have a length of at least 1. This means that the shortest possible mirror will be of length 1, representing a single element (which is its own mirror). A sequence that is a palindrome (the same forwards as backwards) is considered its own mirror and should be included in your computations. For example, if *a1* is `{6, 1, 2, 1, 4, 1, 2, 1, 5}` and *a2* is `{1, 2, 1}`, your method should return `true`. The two occurrences of the mirror might overlap, as shown in the fourth sample call below.

The following table shows some calls to your method and their expected results:

Array	Returned Value
`int[] a1 = {6, 1, 2, 1, 3, 1, 3, 2, 1, 5};` `int[] a2 = {1, 2};`	`hasMirrorTwice(a1, a2)` returns `true`
`int[] a3 = {5, 8, 4, 18, 5, 42, 4, 8, 5, 5};` `int[] a4 = {4, 8, 5};`	`hasMirrorTwice(a3, a4)` returns `false`
`int[] a5 = {6, 3, 42, 18, 12, 5, 3, 42, 3, 42};` `int[] a6 = {42, 3};`	`hasMirrorTwice(a5, a6)` returns `true`
`int[] a7 = {6, 1, 2, 4, 2, 1, 2, 4, 2, 1, 5};` `int[] a8 = {1, 2, 4, 2, 1};`	`hasMirrorTwice(a7, a8)` returns `true`
`int[] a9 = {0, 0};` `int[] aa = {0};`	`hasMirrorTwice(a9, aa)` returns `true`
`int[] ab = {8, 9, 2, 1};` `int[] ac = {5, 7, 1, 2, 9, 8};`	`hasMirrorTwice(ab, ac)` returns `false`

Do not modify the contents of the arrays passed to your method as parameters.

Solutions

1.

Expression	Value
`!(4 > 3.5) || ((6 < 3) && (4 < 9))`	`false`
`99 % 100 + 10 / 4 / 1.0`	`101.0`
`"8" + 3 * 2 + (3 * 25) + 3 * 10 + 3 * 3`	`"8675309"`
`6 / 2 + (14 < 3)`	`error`
`-1 + 11 % 7 * 2 + 1000 + 110 * 3`	`1337`

2.

Array	Final Contents of Array
`int[] a1 = {8, 9};` `mystery(a1);`	`{17, 9}`
`int[] a2 = {14, 7, 1};` `mystery(a2);`	`{22, 8, 1}`
`int[] a3 = {7, 1, 3, 2, 0, 4};` `mystery(a3);`	`{17, 10, 9, 6, 4, 4}`
`int[] a4 = {10, 8, 9, 5, 6};` `mystery(a4);`	`{38, 28, 20, 11, 6}`
`int[] a5 = {8, 10, 8, 6, 4, 2};` `mystery(a5);`	`{38, 30, 20, 12, 6, 2}`

3.

```
Go
```

4.

```
one fish, two fish
blue red, fish fish
red fish, blue fish
```

5.

```
Lois a   Meg a
Lois b
Meg

Lois a   Meg a   Stewie a
Lois a   Meg a   Stewie a   Brian b
Brian Stewie

Meg a
Meg b
Meg

Lois a   Meg a
Lois a   Meg a   Brian b
Brian
```

6.

```
public static boolean isSumArray(int[] array) {
   if (array.length % 3 != 0) {
      return false;
   }

   for (int i = 2; i < array.length; i += 3) {
      if (array[i] != array[i-1] + array[i-2]) {
         return false;
      }
   }

   return true;
}

public static boolean isSumArray(int[] array) {
   if (array.length % 3 != 0) {
      return false;
   }

   for (int i = 0; i < array.length; i += 3) {
      if (array[i+2] != array[i+1] + array[i]) {
         return false;
      }
   }

   return true;
}
```

7. Two possible solutions are shown.

```
public static void printWinner(Scanner input) {
   int max = 0;
   String maxName = "";

   while (input.hasNext()) {
      String name = input.next();
      int counter = 0;
      while (input.hasNextInt()) {
         counter += input.nextInt();
      }
      if (counter > max && counter <= 21) {
         max = counter;
         maxName = name;
      }
   }

   if (max > 0) {
      System.out.println(maxName + " is the winner!");
   } else {
      System.out.println("Everyone busted!");
   }
}
```

```
public static void printWinner(Scanner input) {
   int max = 0;
   String output = "Everyone busted!";

   while (input.hasNext()) {
      String name = input.next();
      int counter = 0;
      while (input.hasNextInt()) {
         counter += input.nextInt();
      }
      if (counter > max && counter <= 21) {
         max = counter;
         output = name + " is the winner!";
      }
   }
   System.out.println(output);
}
```

8.

```
public class Chameleon extends Critter {
     private int count = 0;

     public Action getMove(CritterInfo info) {
         count++;
         if (count % 3 == 1) {
             return Action.INFECT;
         } else if (info.getFront() == Neighbor.EMPTY) {
             return Action.HOP;
         } else {
             return Action.RIGHT;
         }
     }

     public Color getColor() {
         if (count % 3 == 0) {
             return Color.RED;
         } else if (count % 3 == 1) {
             return Color.WHITE;
         } else {
             return Color.BLUE;
         }
     }

     public String toString() {
         if (count % 3 == 0) {
             return "R";
         } else if (count % 3 == 1) {
             return "W";
         } else {
             return "B";
         }
     }
 }
```

9. Four possible solutions are shown.

```
public static boolean hasMirrorTwice(int[] a1, int[] a2) {
    int mirrorCount = 0;
    for (int i = 0; i <= a1.length - a2.length; i++) {
        int count = 0;
        for (int j = 0; j < a2.length; j++) {
            if (a1[i + j] == a2[a2.length - 1 - j]) {
                count++;
            }
        }
        if (count >= a2.length) {
            mirrorCount++;
        }
    }

    return mirrorCount >= 2;
}
```

```
public static boolean hasMirrorTwice(int[] a1, int[] a2) {
    int mirrorCount = 0;
    for (int i = a2.length - 1; i < a1.length; i++) {
        int count = 0;
        for (int j = 0; j < a2.length; j++) {
            if (a1[i - j] == a2[j]) {
                count++;
            }
        }
        if (count >= a2.length) {
            mirrorCount++;
        }
    }

    return mirrorCount >= 2;
}
```

```
public static boolean hasMirrorTwice(int[] a1, int[] a2) {
    // copy/reverse a2 into a3
    int[] a3 = new int[a2.length];
    for (int i = 0; i < a2.length; i++) {
        a3[i] = a2[a2.length - 1 - i];
    }

    int mirrorCount = 0;
    for (int i = 0; i <= a1.length - a3.length; i++) {
        int count = 0;
        for (int j = 0; j < a3.length; j++) {
            if (a1[i + j] == a3[j]) {
                count++;
            }
        }
        if (count >= a3.length) {
            mirrorCount++;
        }
    }

    if (mirrorCount >= 2) {
        return true;
    } else {
        return false;
    }
}
```

```
public static boolean hasMirrorTwice(int[] a1, int[] a2) {
    int mirrorCount = 0;
    for (int i = a1.length - 1; i >= 0; i--) {
        int count = 0;
        for (int j = 0; j < a2.length; j++) {
            if (i + j < a1.length && a1[i + j] == a2[a2.length - 1 - j]) {
                count++;
            }
        }
        if (count >= a2.length) {
            mirrorCount++;
        }
    }

    return mirrorCount >= 2;
}
```