Instructions:
1. Please turn off your cell phones
2. There are 9 questions on this test.
3. You will 3 hours to complete the test.
4. You may not use a calculator.
5. Please make your answers legible.
6. When code is required, write Java code.

1. Expressions. (7 points). For each Java expression in the left hand column, indicate its value in the right hand column. Be sure to show a constant of the appropriate type. For example, 7.0 rather than 7 for a double, Strings in quotes.

A. 5 / 2 + 3

B. 12 % 7

C. 2 + "AT" + 1

D. 10 / 4 + 1.5

E. "DC" + 2 + 3

F. (4 * 2 > 10) || (3 <= 3)

G. (12 / 3 > 0) && !(4 % 2 != 0)
2. Simulation. (2 points each, 8 points total). You are to simulate a method that manipulates an array of integers. Consider the following method:

```java
// 1 < data.length < 8
public static int[] mystery2(int[] data)
{
    int[] base = {2, 7, -1, 2, 5, -3, 10, 7};
    int[] result = new int[data.length - 2];

    int temp1;
    int temp2;
    for(int index = 1; index < data.length - 1; index++){
        temp1 = index - 1;
        temp2 = index + 1;
        if( data[index] > base[index] ){
            result[temp1] = data[temp1] + data[index] + data[temp2];
        } else {
            result[temp1] = base[index];
        }
    }
    return result;
}
```

In the left hand column below is an indication of the elements in an array of integers. In the right hand column indicate the array method `mystery2` would return when the array in the left hand column is passed as the argument to method `mystery2`.

<table>
<thead>
<tr>
<th>Initial Array</th>
<th>Resulting Array after method <code>mystery2</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>{2, 2}</td>
<td>____________________________</td>
</tr>
<tr>
<td>{2, 20, 3}</td>
<td>____________________________</td>
</tr>
<tr>
<td>{-1, 3, 13, 5}</td>
<td>____________________________</td>
</tr>
<tr>
<td>{0, 0, 1, 2, 0, 3}</td>
<td>____________________________</td>
</tr>
</tbody>
</table>
3. Boolean Logic. (5 points) Write a static method that determines if a student taking a class pass/fail has passed the class. The method takes in two pieces of information, a double that is the student's final average in the course and a boolean that is true if the student is an undergraduate student or false if the student is a graduate student. The method shall return true if the student has passed the course, false if they did not pass the course. Undergraduates need a final average greater than or equal to 60 to pass a course, while graduate students need a final average greater than or equal to 70 to pass a course. You must write the complete method, including the method header.
4. Arrays. (15 points) Complete a static method that determines if the elements of an array of integers alternate between positive and negative values. The method shall return true if each positive element in the array is followed by a negative element and each negative element is followed by a positive element. If the elements of the array do not meet this criteria the method shall return false.

Here are some examples:

alternateSigns( {1, -1, 3, -12} ) would return true.
alternateSigns( {1, 1, -3, 1, -2} ) would return false.
alternateSigns( {37, 0, 32, -1, 55} ) would return false. (0 is not a positive value.)
alternateSigns( {-5, 0, -5} ) would return false. (0 is not a positive value.)
alternateSigns( {1} ) would return true. (Array of length 1.)
alternateSigns( {-1} ) would return true. (Array of length 1.)
alternateSigns( {} ) would return true. (Array of length 0.)
alternateSigns( {1, 1} ) would return false.
alternateSigns( {-11, 12, -3, 35, -1} ) would return true.

Complete the method below:

// you may assume data is not equal to null
public static boolean alternateSigns(int[] data){
// more room for the alternateSigns method if needed.
5. 2D Arrays. (15 points) Complete a static method that given a 2 dimensional array of booleans determine which column in the 2 dimensional array has the maximum number of elements equal to true. Return the int index of the column that has the most elements equal to true. If there is a tie between 2 or more columns return the index closest to 0.

For example, if the following 2d array was passed to the method:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Columns 0 and 3 are tied with the maximum number of elements equal to true with 2 elements each. In this case the method would return 0 to indicate column 0 has the maximum number of elements equal to true.

Complete the following method.

```
/* You may assume mat is not equal to null, that mat.length is greater than 0, that there is at least one column per row, and that each row has the same number of columns. mat is not altered as a result of this method.
 */
public static int columnWithMaxTrues(boolean[][] mat){

// more room for the columnWithMaxTrues method on the next page.
```
// more room for the columnWithMaxTrues method if needed.
6. Implementing classes. (15 points) Write a Longhorn class that implements the Critter interface from assignment 11. Instances of Longhorns return the String "B" in their toString() method. They return the color Color.ORANGE in their getColor() method. When Longhorns fight they randomly choose between SCISSORS and ROCK. When Longhorns move they pick a random direction, either NORTH or SOUTH. Initially a Longhorn moves 1 step in the random direction. A new direction is then chosen, but the Longhorn moves a total of 2 steps in that direction before picking a new random direction. After moving 2 steps the Longhorn picks a new random direction and moves a total of 3 steps in that direction. This continues with the length of a leg increasing by 1 each time a new random direction is chosen. The Longhorn constructor takes 0 parameters.

```java
public interface Critter {
    // methods to be implemented
    public int fight(String opponent);
    public Color getColor();
    public int getMove(CritterInfo info);
    public String toString();

    // Definitions for NORTH, SOUTH, EAST, WEST, CENTER,
    // ROCK, PAPER, and SCISSORS
}

Complete your Longhorn class below:
```

// more room for the Longhorn class on the next page.
// more room for the Longhorn class if needed.
7. Strings. (10 points) Complete a static method that returns true if two Strings are different by a exactly one character. The Strings are not necessarily the same length. If the Strings are different lengths each missing character counts as a different character. The Strings may contain characters besides letters.

Here are some examples:

oneDifferent("two", "too") would return true.
oneDifferent("too", "too") would return false. (There are zero differences.)
oneDifferent("to", "too") would return true. (The first two characters of the Strings are the same and the missing third character from the first string counts as the 1 character difference.)
oneDifferent("TOO", "too") would return false. (All three characters are different.)
oneDifferent("to", "tooo") would return false. (There is a two character difference.)
oneDifferent("help", "hlep") would return false. (There is a two character difference.)
oneDifferent("anger", "aner") would return false. (There is a three character difference. The g in anger does not match the e in aner, the e in anger does not match the r in aner, and the r in anger does not have a corresponding character in aner.)
oneDifferent("anger", "angel") would return true.

You will need the length() and charAt(int pos) methods from the String class. Recall that character positioning in Strings, like arrays, starts at 0, not 1. You may also find the Math.min(int, int) method useful. This method returns the minimum of two int parameters.

Complete the following method:

// You may assume s1 and s2 are not equal to null.
public static boolean oneDifferent(String s1, String s2){

// more room for the oneDifferent method on the next page.
// more room for the oneDifferent method if needed.
8. ArrayLists (10 points). Write a static method that takes an ArrayList of Strings and moves all the Strings with a length less than or equal to some given value to the front of the ArrayList.

For example assume the given ArrayList had the following values in it:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;dog&quot;, &quot;mouse&quot;, &quot;cat&quot;, &quot;pig&quot;, &quot;boy&quot;, &quot;girl&quot;, &quot;a&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the given length was 3 all elements of the ArrayList that have a length less than or equal to 3 are moved to the front of the ArrayList. The relative order of elements stays the same. The resulting ArrayList in this case would be:

["dog", "cat", "pig", "boy", "a", "mouse", "girl"]

Recall the following methods for the ArrayList class. For this problem the data type E will be String.

- void add(E value) appends the value at the end of the list.
- void add(int index, E value) inserts the given value at the given position, shifting subsequent values to the right.
- E get(int index) returns the value at the given index.
- E remove(int index) removes and returns the value at the given index, shifting subsequent elements to the left.
- E set(int index, E value) replaces value at given index with new value. The old value is returned.
- int size() returns the number of elements in this list.
- void clear() removes all elements from the list.

Complete the following method:

```java
// You may assume list is not equal to null.
public static void moveToFront(ArrayList<String> list, int length) {

Complete the method on the next page.
```
// You may assume list is not equal to null.
public static void moveToFront(ArrayList<String> list, int length){
9. Arrays. (15 points) Write a static method that takes an array of ints, a minimum value, and a maximum value. The method returns a new array containing all of the elements in the original array that are between the minimum value and the maximum value inclusive. The relative order of the elements in range is reversed in the array that is returned, but unchanged in the original array.

For example if the original array contained the following elements:

\[0, -5, 10, 7, 12, 5, 4, 5, 21, 0\]

and minimum was equal to 5 and maximum was equal to 12 the returned array would be:

\[5, 5, 12, 7, 10\]

Complete the following method:

```java
// You may assume data is not equal to null and that max >= min.
public static int[] reverseInRange(int[] data, int min, int max) {
    // more room for the reverseInRange method on the next page.
```
// more room for the reverseInRange method if needed.