Please complete the textbook sections 1.1 and 1.2 before beginning this assignment.

CS 278
Programming Assignment 1: Truth Tables

Objectives:

- Refresh your Java programming skills
- Master the use of for loops
- Practice using the "for-each" loop structure
- Master the use of one-dimensional arrays
- Practice returning an array reference from a method

Writing Truth Tables

A truth table for a compound proposition with three propositional variables p, q, and r consists of four columns: a column with truth-values of p, a column with truth-values of q, a column with truth-values of r, and a column with truth-values of the compound proposition.

Notice that the columns for p, q, and r are filled according to the procedure given in the textbook. The column for r (the rightmost variable) is filled with alternating T, F, T, F ... The column for q is filled with T, T, F, F, ... (alternating in groups of two). The column for p is filled with T, T, T, T, F, F, F, F (alternating in groups of four).

p	q	r	Compound Proposition
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

For 3 variables, there will be 2^3 (8) rows in the table. The eight blank cells in the fourth column can be filled in many different ways depending on the compound proposition that we choose to evaluate.

If we had 4 variables, there would be 16 rows. If there are n variables, the truth table will have 2^n rows.

Equivalence

Two truth tables are the same when the truth values of propositions are the same for each combination of truth values of variables. The tables are different when the truth values of propositions are different for at least one combination of truth values of variables.

Below are examples of two different truth tables. They have different values in the 3rd row.

Truth table 1

p	q	r	Compound Proposition # 1
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	F

Truth table 2

p	q	r	Compound Proposition # 2
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	F

The Big Picture

For this assignment, you will write a Java program that generates and prints out the truth table with three propositional variables p, q, and r.

You will also write methods that determine the truth value for expressions that use the logical operators ¬ ∧ ∨ and ⟶.

The output must have a clear message before each truth table showing the number of the truth table. Each table must have column headings and the dashes (using hyphens, not underscores).

When you print a truth table, it must have this format.

```
Truth table for ...

p  q  r  proposition
-- -- -- -----------
T  T  T  T
T  T  F  T
T  F  T  T
T  F  F  T
F  T  T  T
F  T  F  T
F  F  T  T
F  F  F  T
```

Note: Printed truth tables must have ‘T’s and ‘F’s in them, not the words true and false.

Take a moment and read the "Java Bytes 1" document that comes with this assignment before you continue.

Let's Break It Down

Part A: JGrasp / Create the first Java program.

1. Log in. Start JGrasp. Create a new Java Program. Save the program as PA1Test.java.
2. Type the basic framework for a Java program:

    - import the Scanner class
    - write a class header
    - write the main method header
    - print "hello world!"
    - match up the curly braces

    Remember that Java is case-sensitive. When you are asked to use a specific name for a class, method, or variable, be careful to type the name exactly as given.

3. Run the program. Debug as needed.
4. You may be saying to yourself "but it's not finished yet!" That's right. Throughout this class, we will use the process of Incremental Programming. Write a small amount of code. Debug and test. Then write the next bit. Debug and test. Each increment is small, yes, but you will find that you get finished with the assignment much faster overall. Also, after each increment, you have programs that you can submit. Even if it's not completely finished, you can still get partial credit.

Part B: Adding Required Documentation

***** Documentation is required in every program for every assignment in this course. *****

1. Add header comments. Header comments are required in every program you write for this class. The following must be included in single-line comments (using //) at the top of the file:
    a) your name
    b) the course and the section you are enrolled in
    c) the assignment this program applies to
    d) the date you last modified the program
    e) purpose of the program

2. Throughout the code, write a few inline comments. Each inline comment must be placed on separate line, before the lines of code that perform the task.

    Examples of inline comments:

```
// input the name and age from the keyboard

// read all of the data from the input file

// calculate the area and circumference

// display the truth table
```

Part C: Adding Static Data

1. Add a public static character array that contains two elements 'T' and 'F', in that order. (See the "Java Bytes 1" document for an example of a static variable.)

2. Add a public static boolean array that contains two elements true and false, in that order.

Part D: Adding "Helper" Methods

Add these methods, in order they are listed, before the main method.

As you add the methods to your programs, you must document each method. The following must be included in single-line comments (using //) immediately before the header of each method:

a) purpose of the method
b) description of parameters
c) description of the return value

Note: Some of these methods will not be called in the finished version of your main method. You must still test them and they will be part of the grading rubric for this assignment.

1. Add a method called buildTableRows. The method will not have parameters.

   The method should create a String array with 8 elements. Fill the array using the process below. When the method is finished, there should be 8 elements in the array, corresponding to the 8 rows of a truth table.

   Return a reference to the String array. (See the "Java Bytes 1" document for an example of how to return an array reference.)
   The process:

   The Strings in the array will be the text needed for the first 3 columns in the truth table (the columns labeled p, q, and r).

   What should be in the array that is created by the method? Let's look at how the array elements correspond to the truth table:

```
                                Truth table for ...
                                p  q  r  proposition
                                -- -- -- -----------
array[0] "T  T  T  "            T  T  T
array[1] "T  T  F  "            T  T  F
array[2] "T  F  T  "            T  F  T
.                               T  F  F
.                               F  T  T
.                               F  T  F
                                F  F  T
                                F  F  F
```

The Strings must include spaces to make the table line up properly. *** This is critical. If you do not have the correct number of spaces in each string, you will not be able to solve the rest of this assignment correctly. ***

You must use 3 nested "for-each" loops to generate the Strings. See the Java Bytes 1 document for how to write a for-each loop.

If you write 8 assignment statements, you will receive 0 credit for this method.

Increment: Add statements to your main method to test the buildTableRows method and print the array that is returned. The test output should look like this:

```
T   T   T
T   T   F
T   F   T
T   F   F
F   T   T
F   T   F
F   F   T
F   F   F
```

Once you have verified correctness of the method, delete the test statements you added to the main.

Take a moment and read the "Java Bytes 2" document that comes with this assignment before you continue.

2. Add a method called booleanToChar. The method should accept one parameter, type boolean. The method should return a char value, either 'T' or 'F'.

   Increment: Remove the print statement with "hello world! and add statements to your main method to test the booleanToChar method. Once you have verified correctness of the method, delete the test statements you added to the main.

3. Add a method called charToBoolean. The method should accept one parameter, type char. The method should return a boolean value. If the parameter value is 'T', the method should return true. If the parameter value is 'F', the method should return false. If the parameter is any other character, the method should throw an invalid argument exception. Here's the syntax:

```
if ( the parameter isn't a 'T' or an 'F' )
    throw new IllegalArgumentException("Parameter must be 'T' or 'F'.");
```

   Increment: Add statements to your main method to test the charToBoolean method. If you call the method with any character other than 'T' or 'F', your program should break and display the illegal argument message.

   Once you have verified correctness of the method, delete the test statements you added to the main.

4. Add a method named **not** that will accept one parameter. (See the code on the next page.)

   I'll give you the code for this one. Examine the code carefully. You will need to use the same ideas for the **and, or, and implies** methods. Pay particular attention to the following:

   - The parameter type is char.
   - The method returns a char value. The return value should be the negation of the parameter. If the parameter is 'T', the method should return 'F'. If the parameter is 'F', the method should return 'T'.
   - The method passes the parameter to the charToBoolean method (see step 2 above), then takes the boolean value returned by charToBoolean, negates it, and passes it to the booleanToChar method.
   - In the last step, the method returns the result of calling booleanToChar.

   We have to have a way to deal with parameters that are not 'T' or 'F'. It's called exception handling. (See the Java Bytes 2 document for information on exception handling.)

   The charToBoolean method can potentially throw an Illegal argument exception. We don't want the program to break. But, we do want to let the user know that there was a problem.

   > Exception messages should contain useful information:
   > 1. What problem occurred?
   > 2. Where did the problem occur? (at the very least give the method name)
   > 3. How can this problem be prevented / avoided?

```
public static char not ( char c ) {
   boolean value = false;
   try {
      value = charToBoolean( c );
   }
   catch (IllegalArgumentException e) {
      System.out.println("An error occurred in the not method.  "
                         + e.getMessage()
                         + " Defaulting to false.");
   }
   return booleanToChar( !value );
}
```

   Increment: Add test statements to the main. Call the **not** method at least 3 times: with 'T', with 'F', and some invalid character. Print the results with meaningful text.

   Once you have verified correctness of the method, delete the test statements you added to the main.

5. Write the method for **and** (logical operator $\wedge$). Accept two char parameters and return a char. Be sure to use a try/catch to handle the possible illegal argument exception from charToBoolean.

6. Increment: Add test statements to the main. Call the **and** method at least 6 times: with 'T' and 'T', 'T' and 'F', 'F' and 'T', 'F' and 'F', with some invalid character for the first parameter, and with some invalid character for the second parameter. Print the results with meaningful text.

7. Once you have verified correctness of the method, delete the test statements you added to the main.

Part E: Planning ahead

Want to get ahead in this class? Have some extra time? Want to earn an A? Here's how:

1. Write the method for **or** (logical operator $\vee$)**.** Accept two char parameters and return a char. Be sure to use a try/catch to handle the possible illegal argument exception from charToBoolean.
2. Write the method for **implies** (logical operator $\longrightarrow$)**.** Accept two char parameters and return a char. Be sure to use a try/catch to handle the possible illegal argument exception from charToBoolean.

Just add the methods to PA1Test.java. It's okay if they are in the PA1Test.java file that you submit for PA 1. These methods aren't part of the grade for PA 1 but they will be needed (and graded) in the next assignment.

Part F: Now for the final version of the main

1. In the main method, be sure to remove any test statements left over from earlier steps.
2. Create a string array.
3. Call buildTableRows and assign the return value to your string array.
4. Print the heading for a truth table (4 lines of output). Substitute the text "not(p) and q" for the "..." and "proposition".

5. Write a for-each loop to go through the string array. For each element in the string array, print the array element followed by the value of the proposition not(p) and q for the variable values in that row.

   The truth table should have 8 rows. Here's what the beginning of the truth table should look like:

```
Truth table for not(p) and q

p  q  r  not(p) and q
-- -- -- ------------
T  T  T  F
T  T  F  F
.
.
.
```

6. Compile. Debug and test. Repeat as needed until it works correctly.

Submit: PA1Test.java on Canvas