Homework 3 CSC 116 Fall 2019

Due ***November 9 11:45 pm***

Note: This assignment can be done in groups of two. You can work with maximum of one other student. You are free to work on your own. (If you are looking for a group member you can post on Piazza to find other searching members).

Note: If you are submitting as a group, only one member from each group should submit the files. Also, you need to make sure that both names are included using @author in java documentation of each file.

This assignment includes 2 incomplete Java files which must be completed and submitted by the due date. Each file requires you to complete methods which are tested in the main method for each program. Sample output is included below.

How I am making your life difficult this time around: For this assignment, I want to allow you to use any and all knowledge of Java that you may have and not limit you to what we've covered already. There are a couple of restrictions which you can read in the method documentation. I also want to add the general rule of not attempting to fool the tests. With enough if statements, you can fake any calculation and that's not what I want to see here.

For this assignment, I wanted to add a few more examples of how the things we have learned in class apply to broader computer science scenarios. A few methods will require some context that I hope to provide here.

Booleans.java

A struggle I've seen in students every semester is the understanding of how to properly use booleans as mathematical values. So, in this program, there are three methods which (I hope) force you to treat them as such.

xor(boolean a, boolean b)

- This method expects you to implement the XOR (exclusive OR) operation without using Java's built in XOR operator.
- XOR behaves like the following:
    - false XOR false = false
    - false XOR true = true
    - true XOR false = true
    - true XOR true = false
- xor() is supposed to return 1 if the operation returns true and 0 if it returns false.
- So, therefore:
    - xor(true, true) should return 0
    - xor(true, false) should return 1
    - xor(false, true) should return 1

    - xor(false, false) should return 0
- How you choose to calculate these values is up to you. There are infinitely many ways. One of the simplest is: $A \cdot \overline{B} + \overline{A} \cdot B$ ((a AND NOT b) OR (NOT a AND b)).
- The completion of this method is worth 10 points.

halfAdder(boolean a, boolean b)

- This method is meant to demonstrate how Boolean operations are used in computers to perform actual math.
- For this method, you are required to return a String comprised of two integers separated by a space.
    - halfAdder(false, false) should return "0 0"
    - halfAdder(false, true) should return "0 1"
    - halfAdder(true, false) should return "0 1"
    - halfAdder(true, true) should return "1 0"
- This requires two formulas:
    - The left digit in the return string
        - a AND b
    - The right digit in the return string
        - a XOR b
- These formulas have the interesting property of being able to display the number of true inputs in binary (10 is 2 in binary, 01 is 1, 00 is 0).
- https://en.wikipedia.org/wiki/Adder_(electronics)#Half_adder
- This method is worth 10 points

fullAdder(boolean a, boolean b, boolean c)

- Addition is performed in computers by linking full adder circuits together.
- Full adder circuits can be constructed using half adders, which is what we'll do in this method.
- fullAdder(a, b, c) can be constructed as the following:
    - output1 = halfAdder(a, b)
    - high1 is true if the left character from output1 == 1
    - low1 is true if the right character from output1 == 1
    - **(the above steps have already been done in your method)**
    - output2 = halfAdder(low1, c)
    - high2 is true if the left character from output2 == 1
    - low2 is true if the left character from output2 == 1
    - output3 = ""
    - if high1 OR high2 is true,
        - output3 = output3 + "1 "
    - otherwise,
        - output3 = output3 + "0 "

        - if low2 is true
            - output3 = output3 + “1”
        - otherwise,
            - output3 = output3 + “0”
        - return output3
    - Outputs:
        - fullAdder(false, false, false) should return “0 0”
        - fullAdder(false, false, true) should return “0 1”
        - fullAdder(false, true, false) should return “0 1”
        - fullAdder(false, true, true) should return “1 0”
        - fullAdder(true, false, false) should return “0 1”
        - fullAdder(true, false, true) should return “1 0”
        - fullAdder(true, true, false) should return “1 0”
        - fullAdder(true, true, true) should return “1 1”
    - This method is worth 10 points

forToWhile(int iNaught, int iFinal, int di)

- This method will output the values of a loop iterating from iNaught to iFinal, increasing/decreasing by di each time.
- Your job is to create the same output using while loops (or do/while loops).
- Note that the output of both loops should be present in the output of your program
- This method is worth 10 points

integerList(Scanner scan)

- This method will use the given scanner to read in input from the user until they enter -1.
- Read in every integer (Except for -1) and ignore every non-integer.
- Return a String containing all integers entered separated by commas.
- This method is worth 10 points

Sample output:

```
$ java Booleans
Results of xor method:
a b|a XOR b
---|---------
0 0|0
0 1|1
1 0|1
1 1|0

Results of half adder method:
a b|h l
---|---
0 0|0 0
0 1|0 1
1 0|0 1
1 1|1 0

Results of full adder method:
a b c|h l
-----|---
0 0 0|0 0
0 0 1|0 1
0 1 0|0 1
0 1 1|1 0
1 0 0|0 1
1 0 1|1 0
1 1 0|1 0
1 1 1|1 1

Using for loop:
1 2 3 4 5 6 7 8 9 10
Using while loop:
1 2 3 4 5 6 7 8 9 10
Using for loop:
0 -2 -4 -6 -8 -10
Using while loop:
0 -2 -4 -6 -8 -10
Using for loop:
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
Using while loop:
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

Lists from preset scanners:
---------------------------------
Give me some input: Your list is ""
Give me some input: Give me some input: Your list is ""
Give me some input: Give me some input: Your list is "3"
---------------------------------

List from user:
---------------------------------
Give me some input: hello
Give me some input: my
Give me some input: 1
Give me some input: name
Give me some input: 2
Give me some input: 3
Give me some input: is
Give me some input: spencer
Give me some input: 4
Give me some input: -1
Your list is "1, 2, 3, 4"
---------------------------------
```

Files.java

Much of the advanced operations of computers is little more than reading or writing to files and doing some calculation on the input. With this program, I hope to illustrate some of the basics of file manipulation.

convertFile(String inputFile, String outputFile)

- This method expects you to take data from an input file, transform it, and print it to an output file.
- The inputFile is a String which represents a file name. This file is filled with integers.
- Your job is to read every integer from that file and print it as a character to the output file.
- Look at java.txt and expected_java_output.txt for a clear example of what this means.
- Hint: if I have integer i, I convert it to a character with char c = (char) i;
- If a FileNotFoundException is thrown when creating the Scanner or PrintStream, you are expected to throw an IllegalArgumentException
- This method is worth 10 points

reportNamesStartingWith(String start)

- In this method, you are expected to create a Scanner over the file people.txt.
- This file has data about 50 people. Each line takes the form: name: <name> age: <age> height: <height>. However, the information is not always in that order.
- You are expected to read the whole file and print the names which start with the given String. Only print the names, no other information.
- You do not have to be case-insensitive. Print the name separated by spaces with a new line at the end. A trailing space is okay.
- If the Scanner throws a FileNotFoundException, just use the line:
    - return;
    - to immediately exit the method.
- This method is worth 10 points

wordCount(String inputFile)

- In this method, you are expected to create a Scanner over the file with the given name.
- You are expected to count the total amount of tokens (“2” counts as a token, you don’t need to ignore numbers or anything) in the file and return it.
- If the file does not exist or cannot be read, you should return 0.
- This method is worth 10 points.

toDouble(String s)

- In this method, you are expected to return the double represented by s.
- For example, if s is “3.5” you should return 3.5

- If s is not a double, you should return Double.NaN. This is a special value which means “not a number.”
- It’s okay to return 3.5 if s is “3.5 the rest is not a number”
- This method is worth 10 points

createFile (String filename)

- In this method, you are expected to create a blank file with the given filename.
- You should return true if the file was successfully created and false if it was not OR the file already existed. If the file already existed, you should not overwrite it.
- This method is worth 10 points.

Sample output:

```
$ java Files
java output files equal
ncsu output files equal

Names starting with "S":
Samira Stefan Spencer

Word counts:
notafile.txt: 0
frankenstein.txt: 384

Parsing doubles:
"": NaN
"Hello": NaN
"345.6": 345.6

Creating a file:
File created!
File could not be created.
```

RUBRIC

This homework is worth a **total of 100 points**. There are **10 methods** to complete, each of which is worth **10 points**.

- Each method will be tested *by a computer program*. Passing these tests will result in 5 points. Partial credit may be awarded if the method passes some tests and not others.
- Each method will be inspected *by a human being*. Passing inspection will result in the remaining 5 points. Partial credit may be awarded, if, for example, the method fails all tests but is mostly correct. Alternatively, if the method passes all tests but uses some method which has been disallowed by the method description, these points would be lost.
- Points can be taken away for reasons including but not limited to

- Not following implementation instructions for certain parts of the assignment
- Attempting to falsify program output
- Attempting to fool tests
- Compilation errors

FILES TO SUBMIT

Booleans.java

Files.java

If you are submitting as a group, only one member from each group should submit the files. Also, you need to make sure that both names are included using @author in java documentation of each file.