EECS2030: LAB 2 Due: Jan 30th, 2022- 11:59 pm

The purpose of this lab is to ensure that you practice

A) designing recursive algorithms.

B) testing your code thoroughly using JUnit test cases.

C) generating the existing javaDoc into an html format.

1. Setup

Please download Lab2.java that is attached to this description.

- Open `eclipse`.
- Click on *`File`* and select *`Import`*.
- Choose *`Existing Projects into Workspace`* and click *`Next`*.
- Click on *`Select Archive File`* and then *`Browse`*. Find `Lab2.zip` and click *`Finish`*.

You should see two files, one is called Lab2.java and one Lab2Tester.java.

2. Important Notes:

You are not allowed to use any regular expressions, loops, or any methods such as "contains", "replace" and so on to solve the problem. In other words, the problem should purely be solved by **recursion,** and therefore any method that lets you get away from designing a recursive algorithm should be avoided.

To practice testing, we only provided a set of **incomplete** test cases. You should make sure that you add enough test cases to the tester that tests your code thoroughly. Please have a look at the tester code `Lab2Tester`, and add more test cases to test your code thoroughly. To do this, you can copy one of the methods, change the name of the method to avoid having a duplicate method's name and change the body of the method to test your code with your selected input.

3. JavaDoc generation

The javaDoc has been written for you. All you need to do is to generate it as an HTML file to make it easier for navigation. For this, right click on `Lab2.java` `->` select `export -> javaDoc -> Next`. It will ask you for the location in which you want to store the documentation. Enter the path and then click on `Finish`.

If you look at the location in which you stored the documentation, you'll see there is a file called `index.html`. Clicking on this file, shows the documentation of the project in your browser.

4. Programming Task 1

For this task, we are asking you to implement a recursive method that finds the sum of n consecutive integer number, starting from integer `start` and ends with integer `end`.

For example, if start = 0 and end = 5, we expect the method to return 15 (i.e. 0+1+2+3+4+5).

The name of the method is `sum` and the header of the method was written for you in Lab2.java.

5. Programming Task 2

For this task, you are expected to implement a recursive function that creates and returns a string of length n using the given character.

For example, if the given character is *, and n = 5, the output should be `*****`.

Please note that a string with length zero is also possible.

To concatenate to a string, the simple '+' operator should be used.

The name of the method is `makeString` and the header of the method was written for you in Lab2.java.

6. Programming Task 3

For this task, you should implement a recursive function that gets 3 parameters and return a string that is made of the first two input parameters repeatedly. This means the output contains the first input followed by the second input, followed by the first input again, …..The number of repetition of the input strings is specified by the third argument of the method.

Here are some examples:

`interlace("Hello ","World ", 0)` returns `""`

`interlace("Hello ","World ", 1)` returns `"Hello "` `interlace("Hello ","World ", 2)` returns `"Hello World "`

`interlace("Hello ","World ", 3)` returns `"Hello World Hello"`

The name of the method is `interlace` and the header of the method was written for you in Lab2.java.

7. Programming Task 4

In this task, you are required to write a recursion that gets a string and two characters and returns the substring that is enclosed in two given characters. You can assume that the given string includes only one instance of each enclosing characters.

For example, where the input string is `This is [quite} an example!` and the first enclosing character is `'['` and the second is `'}'`, it should return `quite`.

The name of the method is `getSubstring` and the header of the method was written for you in Lab2.java.

8. Programming Task 5

For this task, you need to write a recursion that converts a positive integer to its binary equivalence. This method gets one integer input.

A binary form of an integer is calculated by repeatedly dividing the integer number (and later, its quotient) by 2. You keep doing the division until the quotient is zero. The reminders of the divisions, from the last division to the first, form the binary representation of the integer number. Let's see an example in which 23 is converted to a binary number. R stands for the remainder.

23 / 2 = 11 R 1

11 / 2 = 5 R 1

5 / 2 = 2 R 1

2 / 2 = 1 R 0

1 / 2 = 0 R 1

Therefore, the binary representation of 23 is "10111".

The name of the method is `decimalToBianry` and the header of the method was written for you in Lab2.java.

9. Submit

You only submit one file that is called Lab2.java via eClass by clicking on the lab link.

You do not need to submit your tester or HTML files.