CS 110 Lab Sheet 7

Lab cycle starting 13th November 2019, Normal Deadline two weeks from the start of that lab cycle.

General Comment

All these tasks are about writing methods. But as well as the methods, you will have to write some code to test them. So you will also need a main method that calls your methods with different arguments to make sure they work.

Stage 1: Simple Method

Write a method that returns the square of a number. It should start like this:
```
public static int cube(int num) {
```
though you are free to change the name `num` to whatever you like.

Stage 2: The isCube method

Write a method that takes two integers as arguments and returns a boolean result. The result should be true if the first argument is the cube of the second and false otherwise. You might want to try to use your method from Stage 1 and call it from within your new `isCube` method.

Stage 3: Boxes of *

Write a method that takes two integer arguments, representing length and height, and prints out a rectangle of * characters that is length wide and height deep. If you still have your solution to Stage 4 on Lab Sheet 3, this should be easy!

Stage 4: Boxes of * and Do One Thing

It's not good practice for a method to do more than one thing - as soon as you say the word 'and' when you are explaining what a method does, then - probably - you should be writing more than one method - now go and read the description of what the method in Stage 3 above should do….

Having done that, split your method from stage 3 into two methods:

The first should read in one integer - it should reject anything that's not an integer, and it should insist the integer is > 0. It should return the integer that it read in, and it should have at least one parameter which is the string it will print out asking the user to enter an integer - so, for example, you can say what the integer is for ("Enter an integer value for the height of the box"). If you want, it can also have another parameter maxVal that sets a maximum possible value for the integer that is entered - this will come in handy to stop users entering silly values (e.g. 1000 000) for the height of the box that will be printed. And maybe you could do something about negative input at the same time.

The second method should accept two integers - the height and width of the box - and print the box out.

Now write a `main` method that calls your first method twice to read in the height and width; and the second method once to print it out.

Stage 5: Comment Properly

Readers of code need to know what methods do - one place that it makes sense to write relatively long comments is at the start of methods. Go back and write block comments /* … */ in front of all the methods you've written so far, that explain what the methods do - they should say what the parameters represent, and what is computed and returned, or otherwise output. They should not say how the methods work - just what they do.

Stage 6 Assessed Task: Roman Numerals

Write two methods that deal with Roman numerals. The first should take either a string consisting of one character, or (better, easier) a character (`char`) and convert it to it's numeric value. Either:

```
public static int romanNumeralToInt(String romanNumeral)
```

or

```
public static int romanNumeralToInt(char romanNumeral)
```

The values of Roman numerals are:

I = 1
V = 5
X = 10
L = 50
C = 100
D = 500
M = 1000
(all other letters should return zero)

Once you've done this, write another method that uses your first one to convert a sequence of Roman numerals represented as a string into their decimal value. For example
VI = 6; CLX1 = 161, MMXIII = 2013

If you don't use two methods AS WELL AS the 'main' method this task will not be considered correct. Also you must comment your methods properly, as described in Stage 5.

Hints: Use a switch statement in your first method.
You might find these methods useful:

- The string method `substring(x,y)` which returns the part of the string starting at character x and ending at character y-1. So if a string value is equal to "banana split" then value.substring(0,6) is "banana".
- The string method `length()` which tells you how long a string is
- The string method `charAt(x)` which returns the character (char) at location x

There is no need to worry about things like IV representing 4 instead of using IIII, or IX representing 9 instead of VIIII (called subtractive notation) - just add up the character values (the Romans didn't consistently use subtractive notation themselves so I don't see why we should:-).

Challenge Task

Extend your Roman numerals code so that it can add up two strings that represent roman numerals and then convert the result to decimal. Converting them first and then adding them does not count! To do this you might find this page helpful:

http://turner.faculty.swau.edu/mathematics/materialslibrary/roman/

Go to the section marked Addition and skip the first stage in the algorithm (unless you've been crazy enough to include subtractive notation in Stage 5).

Challenge Task

A recursive method is one that calls itself. The definition of the factorial operation n! is:

0! = 1
n! = n * (n-1)!

Write a method that recursively calculates n! for arbitrary n.

The definition of a Fibonacci number is:

Fib(0) = 1
Fib(1) = 1
Fib(n) = Fib(n-1) + Fib(n-2)

Write a method that recursively calculates Fib(n) for arbitrary n.
Note: for those of you who know about such things, don't worry about anything to do with efficiency!