CS206 Introduction to Data Structures

Lab 7b

Recursion

This lab is just implementing several recursive functions. Many of these could be better and more easily written using iteration. Do it recursively anyway. Most of these questions have good solutions posted in the lecture notes for Friday, Nov 6. I encourage you to try to write these functions before looking at the notes.

First, recall these functions

```
    public void badRecurse(int c)
    {
                System.out.println("A" + c);
                badRecurse(c-1);
    }

    public void goodRecurse(int c)
    {
            System.out.println("B" + c);
                if (c<=0) return;
                goodRecurse(c-1);
    }
```

Enter these into VSC and run each. For instance in a main method

```
            badRecurse(5)
```

What happens with each? Why?

The why for badRecurse is that it lack a base case to stop the recursion.

Write a recursive method that simply prints out the numbers 1..10 (Hint you can do this by changing only one line of goodRecurse.

```
Write a recursive method to do as described below:
/**
 * A recursive function to add two positive numbers
     * @param num1 one of the numbers
     * @param num2 another number
     * @return the sum of the two numbers
     */
    public int rAdder(int num1, int num2)
```

Write a recursive method to do the following:
```
/**
     * Implement multiplication recursively using addition
     * For example, given the args 7 and 4 write a recusive function
     * that computes 7+7+7+7
     * @param i1 a number
     * @param i2 another number
     * @return i1*i2
     */
    public int multiply(int i1, int i2)
```

Write a recursive method to do the following
```
/**
     * Compute the integer part of the log of the value
     * @param i1 the logarithmic base
     * @param i2 the number to compute the log of.
     * @return baseNlog(2, 1100)==>10, baseNlog(10,1000) ==>3,
baseNlog(10,9999) ==> 3, baseNlog(1,9)==>0, baseNlog(4,0) ==> 0.
Watch out for base cases, there are a lot of them for this problem.
For instance, the base 1 log is undefined, so the function returns 0.
     */
    public int logNbase(int base, int value)
```

Write a recursive function to compute the Nth fibonacci number. Recall that Fibonacci numbers are:1,1,2,3,5,8,13,21, .. They are defined as f(n) = f(n-1) + f(n-2). That is the nth fibonacci number is equal to the sum of the n-1 and n-2 fibonacci numbers. So for instance, 21=13+8. Special cases here are that f(1)=1 and f(2)=1. There is a trick to this recursive calculation. Essentially you need to create a private recursive “helper function”. The public function calls this helper who actually does the recursive calculation.

Finally, write the following function
```
/**
     * Write a recursive function to add all the values in the array
     * Hint, this method should not be recursive.  Rather make a
     * private recursive function and call that from here
     * @param array
     * @return the sum of the numbers in the array
     */
    public int addArray(int[] array);
```

I will not collect any of this. I will simply assume that you have worked through and understand how all of these work and that you can write variations on these.