The Magic of Recursion

Due date: by the end of the recitation session

Introduction

Beginners often have trouble figuring out how exactly recursive functions work. Today, you will be working with fairly short recursive functions and try to investigate the magic of recursion. It is not really magic once you understand how it works, but it is not a bad idea to think of it as magic at the beginning.
The main idea of recursion is to solve a large problem by reducing it to smaller and smaller version of it and then finally solving a trivial problem. (It is to some extent similar to mathematical induction, but you may not be familiar with that yet.)

Worksheet: https://goo.gl/QpazNH

Part 1

In this part, you will be working with the following recursive function. You may be tempted to write it in a program and try to run it - DON'T! (If you need a motivation, remember that on an exam or an interview, you will need to read and understand code without running it - might as well practice for it now.)

```
public static int fun2 ( int n1, int n2) {
  if ( n2 < 0 )
    return -1;
  if ( n2 == 0 )
    return 0;
  return n1*n1 + fun2 ( n1, n2-1);
}
```

Figure out what this function does and write its iterative equivalent (a function that performs exactly the same computation, but uses loops instead of recursion). Before you jump into the deep end of the pool and try to answer those questions directly, answer the simpler questions to try to establish what really happens when these recursive functions are called. The questions are listed here for your reference. Answer all of the questions on the attached worksheet.

Consider the following function calls to `fun1` and `fun2`.

`fun2(3,-2)`　　　　`fun2(0,3)`　　　　`fun2(3,5)`

1. For each specific call, answer the questions below.
    a. What is the return value?
    b. How many times (including the initial call to the function) is the function called? Write down each of these calls (they will differ by the parameters passed to the function) in order that they occur.
2. Write down the mathematical expression that is computed by the function? Try to generalize it to a formula that is expressed in terms of n1 and n2.
3. Write an equivalent function using iterations (not recursion). Don't use the formula from questions 2 to simply return the final answer.

Part 2

You can use PythonTutor website (yes, I know you are working in Java) to visualize code execution in several different languages.
Follow the link to the two recursive functions on PythonTutor website: `https://goo.gl/Wfyp4C`
(it tends to be slow to start, so be patient, if it fails, just try again by clicking Visualize Execution).
Use the Forward button to execute the code one line at a time. You can also use the Back button to go backwards (this is something that does not work in a debugger, unfortunately, but for good reasons).

As you are stepping through the code, there are boxes that appear on the right hand side. Those boxes symbolize the stack frames created for each function on the program's stack as the program is running. They contain storage for all of the local variables and parameters. They also contain additional information that allows the program to "jump" from one place in the code to the next.

Investigate the two functions listed in the example and answer the questions on the worksheet (they are listed below for your reference).

1. How many stack frames are created to complete the execution of the call to `fun3`?
2. What value is returned by each recursive call to `fun3` (each call has its own stack frame)? List the value of the parameter together with the return value.
3. What does `fun3` do?
4. In each stack frame, there is a number next to the name of the function (right after the colon). What does this number represent?
5. Step through the execution of recursive calls of `fun4`. What does this function do?

Part 3

Some of you may be familiar with a website called CodingBat. It provides coding problems in Python and Java with live testing.
Let's see if you can solve some of the problems related to recursion.
The website tests your code using many inputs. Try to write code solutions to the following problems. Once you have the correct version (that passes all the tests) or an almost correct version (that might be failing some of the tests), take a screenshot of the screen and paste the image in the worksheet.

Here are some problems that you should look at. You do not need to solve all of them! Concentrate on two problems (other than the first warm-up problem) and spend time as a group to try to come up with the best solution that you can.

1. Bunny Ears, http://codingbat.com/prob/p183649 - this is just a warm up exercise, you can see its solutions on the website.
2. PowerN, http://codingbat.com/prob/p158888
3. ChangeXY, http://codingbat.com/prob/p101372
4. Pair *, http://codingbat.com/prob/p158175
5. Count 7, http://codingbat.com/prob/p101409
6. Cout 8, http://codingbat.com/prob/p192383 - this is related to the above, but with an added twist.