CSCE 222 [Section 502, 503] Discrete Structures for Computing
Spring 2017 – Hyunyoung Lee

Extra Credit Homework Programming Assignment

Due date: You will submit this homework only electronically. Please create a separate directory named *yourLastName-yourFirstName-EC-hw1*, put your program[1] and your README file in the directory, zip it, and turn in the zip file to the correct link on eCampus. Electronic submission of your zip file is due on **Friday, 3/24/2016 before 11:00 a.m.** on `http://ecampus.tamu.edu`.

This is an extra credit homework, and is worth 40 points toward your homework total. Pick one of the following programming languages: C, C++, Java, or Python (if you would like to choose any other language, you need to get the permission from me in advance). Implement the following two *recursive* functions, the Ackermann function and the GCD (greatest common divisor) function using the Euclidean algorithm.

1. (20 points) The GCD function accepts two positive integers as the arguments (a, b) and returns a positive integer. The algorithm for the GCD function is given as:

```
function gcd(a, b) // initially, a and b are positive integers
  if (b == 0)
     return a;
  else
     return gcd(b, a mod b);  // mod is the modulo operator
```

2. (20 points) The Ackermann function accepts two nonnegative integers as the arguments (m, n) and returns a positive integer (slide #17). Refer to the lecture slides "recursive.pdf" for the definition of the Ackermann function.

Requirements are as follows.

1. Output the values of the current arguments whenever the function is invoked (one way of doing it is by placing an output statement in the very beginning of the function body, that prints out the argument values before you do anything with the argument values). For example, the output of the Ackermann function should look similar to what I have in the slide #19 (without the ellipsis!).

2. Count how many times the function is invoked, including the initial invocation from the main() function, until a final answer is returned. One (easiest) way of doing it is to use a global (static) variable that is initialized to zero for each instance of function invocation in the main, and increment it by one in the beginning of the function body. For example, `A(1,1)` will result in four invocations of the function `A()` because `A(1,1) => A(0,A(1,0)) => A(0,A(0,1)) => A(0,2) => 3`, where the underlined invocation is what should be counted.

[1] Name your program file *yourLastName-yourFirstName-EC-hw1* followed by the appropriate extension for the programming language you are using, for example, .cpp if you implemented your program in C++. Furthermore, make sure you put your name and section number as part of the head comment in your program.

3. Have one `main` function that tests (invokes) your functions. Thus, you will have one program file that consists of the `main` function and the two recursive functions.

   In particular, try the following sets of arguments to test your functions and produce the outputs.

   (a) The GCD function: invoke it with the following value pairs:
   $(12, 15), (228, 133), (576, 414)$, and $(1071, 924)$.

   (b) The Ackermann function: invoke it with the following value pairs:
   $(1, 1), (2, 2), (2, 3)$, and $(3, 3)$.

4. Last but not least, include a README file that explains how to compile and run your program. Also, include in the README file what the expected output should look like when your program is run.

Have fun!