CS120 Lab Assignment #5
Fall 2012

Loops are simple programming constructs that make it easy for the programmer to make a program do the same thing repetitively. Functions have a similar advantage; they allow the programmer to define a set of commands that can be 'called' by a name. Anytime you want those commands to run you simply use the function name. Without functions you would need to retype the commands over and over again.

You can send information, via 'arguments', to a function and receive results, via 'return value', from a function. One way to think of a function is as a mini-program that does part of the full program's job for you.

Functions can make a program shorter and easier to read, but more importantly, they can make a program easier to write. Functions make programs easier to write because they make it easy to design and write a long program as a series of smaller, simpler functions. Functions also make it easier for programs to be written by a team. Each team member can write separate functions according to some specific criteria and those functions are put together to form the whole program.

Functions must be *declared* before they can be *defined* or used. The declaration of a function defines the function's name, return type, and arguments. For example, the statement:
float sampleFunction(int, float);
declares a function named *sampleFunction* that returns a floating point value and that takes one integer value and one floating point value as arguments.

The definition of a function gives the code that is executed when the function is called. For example, the code:

```
float sampleFunction(int arg1, float y){
      float product;
      product = arg1*y;
      return product;
}
```

defines the function called sampleFunction. (The function happens to return the product of its two arguments.)

A common way to format programs is to put function declarations before main and function definitions after main:

```
float sampleFunction(int, float); // declaration
...
int main(){
      ...
      sampleFunction used one or more times
```

```
      ...
}
...

float sampleFunction(int arg1, float y){  // definition
      float product;
      product = arg1*y;
      return product;
}
```

1 The Lab

For this project you will need to write a short program that *includes three functions*. The program should get two integers from the user, determine the larger value, and print the answer. This is very similar to a previous lab, but in this case the program must be written to include the three functions:

1. A function that asks the user for a natural number (an integer larger than zero). If the user doesn't enter a natural number the function should print a message and request another number. This should repeat until the user does enter a natural number. The natural number is returned by the function.
2. A function that accepts two integers as arguments and returns the larger of the two numbers. If the numbers are equal the function should return 0.
3. A function that accepts one integer as an argument and prints it.

The main program needs to be written to use these three functions so that the program asks for two natural numbers and prints the larger of them - or zero if they are equal.

As usual use `script` and `cat` to combine your code and output and turn it in using `cscheckin`. Remember to test several cases, including the user entering illegal values.