Homework 3: Functions and Line-by-Line I/O

**Assigned**: February 7th, 2018 **Due**: February 14th, 2018

Make a subdirectory "hw3" of your cs240 folder for this assignment. Copy the starter files from /courses/cs240/s18/kamaral/GROUP/hw3 to your hw3 subdirectory.

**Part 1: Writing a Function**

In the mid.c file, write a function outside of the main function that takes three integer arguments and returns an integer. Use whatever names for the function and argument variables that you want to for this function’s definition.

The purpose of this function is to determine which of the three arguments is the median or middle value of the three. For example, out of 9, 13 and 26, 13 is the median value.

The main function should have at least 5 calls to this function with different values.

For each call to this function, the main function should print which arguments were given to it and what the value of the function is on those arguments.

One line of example output:

(26, 9, 13) => 13

Use whatever format you like, as long as it's clear to the grader.

**Part 2: Searching for a character in the lines**

Before starting this part of the homework, read this part in it's entirety and write pseudo code for this problem in snag.txt.

In the snag.c file, you are going to write a function which takes a string argument and a character argument. It should return a truth value (int 0 or 1), 0 if the string does not contain the character, and 1 if the string does contain the character. Do not use a built-in library for this. Again, call this function and its variables whatever you deem appropriate.

The main function of the program should accept a single character followed by Enter. Then, it will read lines until the end of input (we'll call that EOF from now on). It should pass these lines to the function you wrote. If the character is in the line, it should print "FOUND: " and then the full line. If the character is not in the line, it should print "NOT FOUND: " and then the line.  
  
Assume lines of input are no longer than 1000 bytes in length. Also, make sure that your program can accept an arbitrary number of lines of input. Either 5, 10, 90, or 1000 lines of input can be given after the first character and it should still work as intended even if you gave the program more input. Again, signal the EOF with CTRL+D. You should only need one buffer.

Example input:  
e  
Hello, World!  
Hi, World!

Example output: (may be mixed in with input)  
  
FOUND: Hello, World!  
NOT FOUND: Hi, World!

**Important note about fgets!**

We'll actually discuss this Thursday in class. The slides were missing information about the return value from fgets, which is crucial in determining when it reaches EOF.

The return value from fgets is a pointer value. There are two cases for this:

1. fgets successfully reads in a line of input.  
   Return value of fgets: non-zero char\*-typed address value (not NULL)  
   In fact, this return value should be *exactly* the buffer address it was given.
2. fgets gets to the end of input or experiences and error and fails to read a line.  
   Return value of fgets: zero pointer char\*-typed address (NULL)

The way to check for either case is as follows:

if (fgets(buffer, 1000, stdin)) {  
 /\* line of input successfully read \*/  
} else {  
 /\* end of input (EOF) reached or error \*/  
}

Again, this is because addresses are numeric types, it allows us to use them as Boolean values (zero is false, non-zero is true).

**Turning in the Assignment**

* start the typescript
* ls -la the directory
* cat mid.c, snag.c, snag.txt
* compile mid
* run mid
  + Give an example on some input
* compile snag.c
* run snag
  + Give an example on at least 7 lines of input
  + Make sure the example lines have cases where the character is and isn't in the line.
* Exit the typescript

In your hw3 folder, there should be the following deliverables:

* mid.c
* snag.c
* snag.txt
* mid
* snag
* typescript

Make sure to include your name in the comments of the source files.

I do NOT need a hardcopy.