# CS/240/Lab/0

## Q1: Hello

Welcome to the world of C programming! Here you are standing right in front of the gate of a magic kingdom. While there are many great programmers wandering this land, their very first step is always the same: shout `Hello World`! Let's do it in the five following simple steps:

1. Create the program source file: Open the text editor `gedit`, create a new file, name it `hello.c`, and type the following code and save.

   ```
   #include<stdio.h>
   int main() { printf("Hello World!"); }
   ```

2. Open a terminal window and type `ls`. You should see `hello.c` listed right there.

3. Compile it by typing

   ```
   gcc -o hello hello.c
   ```

4. Type `ls` again and you should see the compiled program `hello` in the current directory.

5. Run your program by typing `./hello`

**Twitter**
You are ready for a real task. We are going to implement a series of small programs to act as a Twitter client. They will accept user input in the form of short text messages, prepares them as needed, and sends them to Twitter. Each small program is a *filter* which reads a sequence of character from its standard input and writes back a transformed sequence of characters on its standard output. The implementation will span lab 0 and 1.

## Q2: Counter

Twitter limits the number of characters in a message to 140. Your first job is to implement a filter, which truncates messages that are longer than 140 characters. The counter should read every character of a message from standard input and only print the first 140 characters.

Here are some hints, to help you out:

- Your program should have a loop that invokes `getchar()` and counts characters.
- The end of the message is marked by the EOF value. When `getchar()` returns it, stop reading.
- You can test for EOF with `c == EOF`.
- Printing can be done via `putchar()`. But EOF should not be printed.

**Using libraries**

So far your programs have used several functions that you didn't write. `getchar()` and `printf()` are all part of the *standard library* for C. To include them in your code you only need to include a header file that declares them. Since the C compiler knows where the standard library is, the code for these functions is automatically added to your code. For functions outside of the standard library, you need to do three things. You need to include a header file that holds the declaration of the function. For example:

```
#include "tweetIt.h"
```

You need a library file that includes the definition of the function you are calling. The file may end in `.a`. For example `libtweetit_1.a`. You must tell the compiler to add the library file to the executable program. This is done by adding the `-l` flag followed by the library name when you compile. You also need to tell it where that library is. This is done by adding the `-L` flag followed by the directory the library is located in. A complete example could look like this:

```
gcc -o tweeter tweeter.c -L. -ltweetIt_1 -lcurl -lssl
```

This means compile an executable named `tweeter` from `tweeter.c` and look in the libraries `libtweetIt_1.a`, `libcurl`, and `libssl` for any functions not defined in `tweeter.c`.

**32- vs. 64-bit library**

The lab machines and the virtual machines are running 64-bit and 32-bit environments respectively. Therefore, the libraries will be provided in both versions. As part of the file names, number `32` or `64` indicates their version. If you are on the lab machines, rename `libtweetit64_1.a` to `libtweetit_1.a` using

```
mv libtweetit64_1.a libtweetit_1.a
```

and if you are on the virtual machine, rename `libtweetit32_1.a` to `libtweetit_1.a` using

```
mv libtweetit32_1.a libtweetit_1.a
```

# Q3: Tweet

Let's write a filter to send a message to Twitter! This filter is simple too: its reads the message into a buffer, a 140 characters array, and calls a library function with that buffer as argument.

Hints:

- The library function is `tweetIt(char msg[], int len)`.
- A message can be shorter than 140 characters.

Since the class is large, we have made several twitter accounts to tweet to. You have been provided multiple libraries that tweet to one of the twitter accounts. The library name reflects the twitter account your code will send to. For example using `libtweetIt_1.a` will post to twitter account `CS240_1`.

**Testing your program**

Since the `counter` and `tweet` read the message from standard input, you could test them by typing in a message by hand and pressing `Control-D` to generate the EOF. However, this gets tiresome. Fortunately, you can do this very easily in Linux by using `cat` command and the pipe. You first need to store your message text in a file, say `msg.txt`. Then type:

```
cat msg.txt | ./counter | ./tweet
```

`cat` is a Linux built-in utility that reads a file and writes its content to the standard output. You can also `cat` more than one file by listing them in the arguments. The result is all files con*cat*enated.
The pipe operator, `|`, connects two programs, directing the output of the first to the input of the second. The above command feeds the raw message to the `counter` for adjusting the length and then the second pipe operator feeds the result to `tweet`.

# Turning in

You will be assigned a unique turnin ID sent in an email to your @purdue.edu email address. Go to the auto grader webpage ( `http://mc18.cs.purdue.edu:8080/cs240` ). On the submission web page, enter your unique turnin ID in the box and press "log in". Under "currently open projects" will be listed "lab 0 counter" and "lab 0 tweeter". Click "lab 0 counter", use the file selector to choose your `counter.c` file, then click "submit". The page will tell you how you've done, and your total score. When you're satisfied, click "return to list of projects", then "lab 0 tweeter". Use the file selector to choose your `tweeter.c` file, then click "submit". When you're satisfied, close your browser window. You may resubmit at any time before the due date by the same process.

Lab is due Wednesday, August 29th, before midnight. No late labs accepted.