APS 105 — Computer Fundamentals
Lab 1: The Program Development Cycle in C
Fall 2013

Important note: Before attempting this lab assignment, read the *Getting Started Tutorial for APS 105F Linux Basics and Compiling Software* handout that is available in the "Labs" section of the course web site.

According to the course syllabus, this lab assignment will be marked by the automarker, but no Teaching Assistants will be available in the labs in the week of September 9—13, 2013. The results of marking this lab will be posted to the "My grades" section of the course website, but will not contribute to your course grade. You must use the `submitaps105f` command to electronically submit your solution by 11:59 p.m. on Saturday, September 14, 2013.

Objective

In this lab you will use a text editor and basic Linux commands to complete the program development cycle. In this cycle, you will write and enter a computer program that uses the C programming language, compile and run your program, and use a debugging program to help locate possible errors in your program. You will gain experience using `printf`. The program development cycle that you practice in this lab will be used in the other labs in this course.

Note: This lab assignment can be completed using a computer in the ECF labs. Experienced computer users may be able to complete the lab using their own computer after reading Section XIII of the *Getting Started Tutorial for APS 105F Linux Basics and Compiling Software*. Others are encouraged to simply use the ECF labs for now.

Part 1 — Developing a simple C program

Write a C program that produces the following output. The output must be *exactly* as shown:

```
C uses escape sequences for a variety of purposes.
Some common ones are:
  to print ", use \"
  to print \, use \\
  to jump to a new line, use \n
```

Note that even spaces are very important and must be exact. There are two spaces in front of each of the lines that begin with "to print" and "to jump".
In a terminal window, use a *text editor* such as *pico*, *nano*, *nedit*, *vim* or *emacs* to create a file named `simple.c`. Enter your C program in to this file. In Mac OS X or Windows, you may choose from one of the many available plain text editors, such as TextEdit in Mac OS X or NotePad in Windows.

You may wish to review Chapter 1 of the textbook before you begin working on your program. You are required to document your program with comments. Your comments

should at least include your full name and student number, and a very short description of what the program does.

Part 2 — Compiling your program

After you have created a file named `simple.c`, you should try to verify that it does not contain obvious errors before you compile it. The Linux command to display the contents of a file is `cat`. You may enter the following command at the Linux command line (in a terminal window):

```
cat simple.c
```

If you wish to view the list of files in the current directory, you may enter the following command:

```
ls -al
```

which displays the names of all the files (including hidden files) in the current directory. If you wish to find out the name of your present working directory, you may enter:

```
pwd
```

which stands for the "present working directory." If you wish to change the name of the file `simple.c` to `basic.c`, you may enter:

```
mv simple.c basic.c
```

You should also practice using some other commonly used Linux commands. For example, the command `cd` changes your working directory to the one you wish to work in. The command `rm` removes one or more files (and even directories, if `rm -r` is used). If you are wondering how to use a particular command, you can use the command `man` to find out. For example, the command

```
man ls
```

shows you all the details of using the `ls` command.

After you become comfortable with these Linux commands, you may use the command:

```
gcc -o simple simple.c
```

to compile your C source file, assuming its file name is `simple.c`. The option "`-o`" is used to let `gcc` know that you wish the executable file be called `simple`. If there are compile-time errors, you should try to fix these errors using your editor, and then compile again. If there are no compile-time errors, you can run the executable file by using the command:

```
./simple
```

and see if it produces the desired output.

Part 3 — Using the `gdb` debugger

After your C program produces the desired output, you can try to run the program again using a debugger called `gdb`, which stands for GNU debugger. You can start by using the command:

```
gdb simple
```

You will see the prompt

```
(gdb)
```

which indicates that `gdb` is waiting for your commands. If you type `help`, you will see a list of classes of `gdb` commands. At the prompt, you can type:

```
run
```

which causes `gdb` to run the program. If all goes well, you will see the desired output, and the line:

```
Program exited normally.
```

Now, you can exit the debugger by using the command `quit`. If you forget to include `return 0;` at the end of your program, what will you see in `gdb`? Please find out.

Part 4 — Debugging your program using `gdb`

In order to do more useful things with `gdb`, you must first recompile the C program with the option `-g`. This will cause extra information that will assist the debugger to be included in the executable program. You can use the command:

```
gcc -g -o simple simple.c
```

You may now use `gdb` to debug the program again:

```
gdb simple
```

When you see the `(gdb)` prompt, you may use the command:

```
start
```

to start the execution of the program. After the execution starts, the debugger will pause and display the first line of the program. At this time, you can use the command `list` to see a list of nearby lines of C code. To execute the program one source line at a time, you can use the command:

```
next
```

and when you wish to continue running the remainder of the program without going through each single line one at a time, you may use the command:

```
continue
```

These commands are useful for you to get a feeling of "stepping through" the C source code while it is being executed, one line at a time.

What to Submit

Submit your Lab 1 file `simple.c` using the following command (by 11:59 p.m. on Saturday, September 14, 2013):

```
% submitaps105f 1 simple.c
```

You may check the status of your submission using the command:

```
% submitaps105f -l 1
```

where `-l` is a hyphen followed by the letter *'ell'*.