ECE 177 **Lab 4** **Spring 2019**

Part 1 Purpose: Using the debugger. More programming practice.

1) Create a new project directory (as you do for each week's lab) and change to that directory. Reminder (`mkdir lab04; cd lab04`). Copy the `lab04.c` source file from the web page:

```
$ wget http://web.eece.maine.edu/eason/ece177/lab04.c
```

Also `wget` the files, `libeaster.a` and `easter.h` from the same location. Compile the code using:

```
$ gcc -g -o lab04 lab04.c  -L. -leaster
```

The "`-g`" switch compiles with debugging information. The other extra stuff links in a "library" file that contains the "`easter()`" function. The given code should compile and the calculations are correct, but it contains a number of errors that make it behave incorrectly. What it should be doing is pretty obvious from the code. Your task is to find and fix the errors, so it produces correct results. **Make minimal changes** (for example, do not reformat anything or change the comment text.) If the problem can be fixed by changing, adding, deleting a single character (or a couple characters), then that is the correct solution for the purposes of this lab. A sample of correct output is given below. Your output should look like that *exactly* (including spaces) for the given sample input.

2) Run with debugging by using the "`gdb`" command. A file called "gdbNotes.txt" is on the class assignments page and it should give you enough information about debugging to compile and debug this code. (Note that there is much more to `gdb` than what is in this help file.) In particular in this lab you should learn how 1) set breakpoints, 2) run the code (until it hits a breakpoint), 3) continue running from the current location, 4) restart from the beginning ("`run`" again), 5) stop execution while code is running (using `Control-C`), 6) single step using "`next`", executing one line at a time, 7) step into a function – (not needed for this lab), 8) list lines of code, 9) print the value of variables and 10) stop debugging and return to the command line (`quit`).

3) You should also fix any `printf()` formatting so that the output exactly matches the output below (including any "whitespace" such as blank lines and other spaces).

4) Also fix the date range error.

5) Although the expression `1800 <= startyear & startyear < 1900` seems to work correctly, it contains an error. Find and fix it.

6) After the corrections are made, use a breakpoint on the line that says "`easter(year, &month, &day);`" and debug using a date range of `1898 - 1903`. After stopping at the breakpoint each time through the loop, print the variable "`day`" and discover the maximum value it takes on for this date range. Use the debugger and don't add `print` statements to the code.

When finished, have a TA check your corrections and use of breakpoints, stepping, and looking at variables. Also, explain to the TA what happens if your end year is earlier than your start year.

Doing the above will get you a “C” grade. To increase your grade by one letter, add code which will print the dates of the earliest and latest Easter within the given date range after executing the loop. E.g., “`The earliest Easter is March 26 and the latest Easter is April 17`”. Complete the “C” grade (including checkoff by the TA), before starting for a higher grade.

Your code should produce the following output exactly (including spaces / line spacing):

```
Enter the start year: 1898
Enter the end year: 1903

startyear is in 1800s!

Date of Easter in 1898 is April 10
Date of Easter in 1898 is April 10

Date of Easter in 1899 is April 2
Date of Easter in 1899 is April 2

Date of Easter in 1900 is April 15
Date of Easter in 1900 is April 15

Date of Easter in 1901 is April 7
Date of Easter in 1901 is April 7

Date of Easter in 1902 is March 30
Date of Easter in 1902 is March 30

Date of Easter in 1903 is April 12
Date of Easter in 1903 is April 12
```

Part 2:

To increase your grade by one more letter, write a program that will compute and print a table of the factorials from 1 to 12. The program should use looping (perhaps two loops are required). It should produce the following output:

```
 1          1
 2          2
 3          6
 4         24
 5        120
 6        720
 7       5040
 8      40320
 9     362880
10    3628800
11   39916800
12 479001600
```