CIS1300 Fall 2019

Lab Assignment 3 - *Preparing for Git and Learning from the Midterm*

Instructions

1. In your Debian 10 VM or from your laptop, open a Terminal and secure shell into the `linux.socs.uoguelph.ca` server:

   ```
   $ ssh <username>@linux.socs.uoguelph.ca
   ```

   where you replace <username> with your UoG login username. On the `linux.socs` server, create the following directory tree (if it does not exist already):

   ~/CIS1300 Remember that ~ refers to your home directory
   ~/CIS1300/Assignments
   ~/CIS1300/Assignments/Assignment1
   ~/CIS1300/Assignments/Assignment2
   ~/CIS1300/Assignments/Assignment3
   ~/CIS1300/Assignments/Assignment4

   ~/CIS1300/Labs
   ~/CIS1300/Labs/Lab1
   ~/CIS1300/Labs/Lab2
   ~/CIS1300/Labs/Lab3
   ~/CIS1300/Labs/Lab4

2. Go back to your Debian VM (you can do this by typing `logout` or `exit`). Transfer all your code for Assignment 1 from your Debian VM to the `linux.socs` server using secure copy. For example, if you are in your Debian VM Assignment 1 directory:

   ```
   $ scp *.c <username>@linux.socs.uoguelph.ca:~/Assignment1
   ```

   `*.c` means to transfer all files whose filenames end in `.c`.
   Copy all source code, test files, header files, Makefiles, etc.

   Transfer all your code (as much as you have done so far) for Assignment 2 from your Debian VM to the `linux.socs` server using secure copy.

   Copy any code that you have for Lab Assignments 1 and 2.

3. Return to the `linux.socs` server:

   ```
   $ ssh <username>@linux.socs.uoguelph.ca
   ```

Go to the directory ~/CIS1300/Labs/Lab3.

In this directory create a C source code file called `countLeaps.c`. This program will do the following tasks:

a. Read in 2 command line parameters: `startYear numLeaps` where `startYear` is a number representing a year and `numLeaps` is an integer number.
b. The program will first check that it has the right number of parameters and if yes it will assign them to variables of appropriate type and name.
c. You will then create a `while` loop (and it MUST be a `while` loop) that checks if a year is a leap year starting from the `startYear`, and then incrementing `startYear` by one, and stopping when it has found `numLeaps` number of leap years.
d. When the program finds a leap year it will print it out on its own line. For example,

```
$ ./countLeaps 2000 3
```

would output

```
2000
2004
2008
```

The routine that finds if a year is a leap year will be a function that you write called `isLeap()`. Its declaration is:

```
int isLeap ( int year );
```

It returns 1 if the year is a leap year and 0 if it is not.

You will be compiling and linking in the following way:

```
$ gcc -ansi -Wall -c isLeap.c
$ gcc -ansi -Wall -c countLeaps.c
$ gcc -ansi -Wall countLeaps.o isLeap.o -o countLeaps
```

Put these compilation rules into a `Makefile` so that if you run:

```
$ make countLeaps
```

you will get the executable `countLeaps`. Also add the following to your Makefile:

```
clean:
        rm countLeaps countLeaps.o isLeap.o
```

4. Have the TA check your set up on `linux.socs` and test your `countLeaps` program.

Grading

- CIS1300 directory structure on `linux.socs.uoguelph.ca` = 1
- Working `countLeaps` program = 1
- Working `Makefile` = 1