Homework 2

Due: October 27, 2015 **Points:**200

Linux

All these questions are to be answered using the CSIF systems. If you use some other system, your answers may differ, and we will grade based on the CSIF systems.

1. (*15 points*) Suppose I have a file named *x* in my home directory. File *y* is a (hard) link to this file, and file *z* is a symbolic link to it. I now type:

   ```
   mv x a
   ```

   What is the output of each of the following commands, and why?

   (a) `cat x`
   (b) `cat y`
   (c) `cat z`

2. (*5 points*) Give a one-line Linux command that replaces every blank space with the letter '`X`' in a file.

3. (*5 points*) Give a Linux command that prints the calendar for the month of September in 1752. What is unusual about that month?

C Programming Language

Please do either of the two questions. You must pick one; you cannot do part of one and part of the other. In your submission, state which one you have done.

4. (*44 points*) In MyProgrammingLab, please do the first question in each subsection of the following sections: 6.1–6.2, 7.1–7.3, 7.5–7.6, 7.Extra, 8.1, 8.4. Please click on "Submit" for each answer, so you can see if your answer is correct. If it isn't, try again. We will consider only the last answer you submit. Also, do not submit your answers to SmartSite; simply say you have completed them, and we will get your score directly from MyProgrammingLab.

5. (*44 points*) In the textbook, please do question 4 in the Review Questions section of chapter 6, questions 1, 3, and 4 in the Review Questions section of chapter 7, and questions 1–4 and 8–10 in the Review Questions section of chapter 8.

Programming

6. (*50 points*) The Fibonacci numbers play an important role in biology, mathematics, and other sciences. The first two numbers of the sequence are 0 and 1, and the numbers of the sequence are formed by adding the two previous numbers; so, the first few terms of the sequence are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... Please write a function, called `fib`, that takes a single integer *n* as its argument, and prints the first *n* numbers of the sequence.

   The function must have the following interface:

   ```
   int fib(int n)
   {
       /* your code goes here */
   }
   ```

   Your function is to print the numbers on a single line with one blank separating numbers. Do not have any leading or trailing white spaces, and terminate the list of numbers with a newline.

   If $n = 0$, print a blank line (as the function is to print 0 Fibonacci numbers).

If the argument *n* is invalid, `fib` is to return 0. Otherwise, it is to return 1.

The function you write must be stored in a file called "fib.c". Do not put a `main` routine in this file. We will supply this.

Hint: There is a file that provides an interface (a `main` routine) available on SmartSite; it is called "fibdriver.c". If you download it and compile it with your function (see **Compiling and Executing Your Program** for details on how to do this), you can focus on writing the function. Also, the program "/home/bishop/ecs30/fib" is a copy of the program on the Gradebot that produces the output with which your program's output can be compared, so feel free to use it to test your program. It *only* runs on the CSIF.

7. (*50 points*) Write a C program that reads a string from the standard input as words, and prints each word and its line number on the standard output. Loop until standard input's EOF, then terminate. A "word" is defined to be any contiguous sequence of alphanumeric characters. Use the `fgets` function to read the input a line at a time. Your program should handle lines of up to 100 characters. Don't bother to check for longer lines; you'll fix this in a later program.

   The interface of your program must look *exactly* like this (your input is in bold; what the computer types is in normal font), if you are entering text from the keyboard:

```
Hello, there, my old friend!
1   Hello
1   there
1   my
1   old
1   friend
How are you today?
2   How
2   are
2   you
2   today
I am very well, thank you!
3   I
3   am
3   very
3   well
3   thank
3   you
Goodbye ...
4   Goodbye
```

   First print the line number, with no leading space, then print a tab character and the word.

   If you are reading input from a file (by redirecting it), just print the line numbers followed by the words; you do not need to print the input too.

   If there is no input, print nothing.

   The program you write must be stored in a file called "words.c".

   Hint: The program "/home/bishop/ecs30/words" is a copy of the program on the Gradebot that produces the output with which your program's output can be compared, so feel free to use it to test your program. It *only* runs on the CSIF.

Debugging

8. (*31 points*) The program *calc.c* discussed in class has a bug: a spurious "invalid operator" message after each loop. Find out why and fix it. Place your explanation for the error, and how you fixed it, in the header comment.