# Homework 3

**Due:** November 9, 2015 **Points:**100

***Note about points.*** If you submit this by Wednesday, November 4, at 11:55pm, we will add 20 points to whatever score you receive. If you submit this by Friday, November 6, at 11:55pm, we will add 10 points to whatever score you receive.

## 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.

You ***must*** put your answers in a file called "Linux.txt" or "Linux.pdf". If you need to submit pictures also, put them in files called "Linux*n.ext*", where *n* is a digit and *ext* is an appropriate extension. ***If you call your file(s) anything else, or submit something other than text or a PDF file, we will not grade it and you will get 0!***

1. (*5 points*) How do you determine who is logged into the CSIF system you are using?
2. (*7 points*) Write a UNIX/Linux command to print dictionary words that contain the vowels 'a', 'e', 'i', 'o', and 'u' in that order. The vowels may be upper-case or lower-case.
   *Hint:* The file "/usr/share/dict/words" is a dictionary of English words. The command *grep*(1) may help you.
3. (*5 points*) What is the file "/etc/motd"? When are its contents displayed?

## 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.

If you do question 4, you ***must*** submit a file called "MyProgLab.*ext*", where *ext* is any 3-letter extension. We will not look at the contents of this file; its presence tells us you did the MyProgrammingLab exercises.

If you do question 5, you ***must*** put your answers in a file called "CPL.txt" or "CPL.pdf". ***If you call your file(s) anything else, or submit something other than text or a PDF file, we will not grade it and you will get 0!***

4. (*18 points*) In MyProgrammingLab, please do the following questions: 10233 (first question of 9.2) and 10226 (first question of 9.3). 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. Note the change: exercise 73119 has been dropped.
5. (*18 points*) In the textbook, please do questions 4 and 5 in the Review Questions section of chapter 9. You do not need to use the `is_element` function as question 5 suggests, but you may do so if you like.

## Programming

You ***must*** put your *function* in a file called "getint.c". Do not include a driver (`main`() routine). ***If you call your file anything else, or submit something other than a C program (for example, a text or PDF file containing the code), we will not grade it and you will get 0!***

6. (*50 points*) When you use "`scanf("%d", &n)`" to read an integer, it reads digits from the standard input up to the first non-digit. So, for example, the input "`234`" will cause `n` to be set to 234, and the input "`56xy7`" will

cause `n` to be set to 56, with the next character to be read being '`x`". This sometimes causes problems, because "`56xy7`" is not a number. Your problem is to write a function to fix this.

We define an integer as follows:

- 0 or more leading white spaces, followed by ...
- an optional '`+`' or '`-`', followed by ...
- 1 or more digits, followed by a non-alphanumeric, but not a '`.`' followed by 1 or more digits.

So for example, "`1234`", "` 1234 `", "`1234.`", "` +1234 `", and "` -1234 `" are all integers, and none of "`1234e5`", "`e1234`", "`1234.56`", and "`1234abc`" are.

For the purposes of this function, assume that the next line of input was read into an array, for example by *`fgets`*, before your function is called. Your function is to have the interface

```
int getint(char *inputline, int *value, char **nextchar)
{
    ...
}
```

The character pointer `inputline` points to the input line. Generally, your function is to return the number read in the location that `value`, and the address of the first character *after* the last digit of the integer is stored in the location pointed to by `nextchar`.

More specifically:

(a) If any of the arguments to *`getint`* are invalid (see the first hint), the function returns $-1$.
(b) If *`getint`* finds an integer is the first thing in the `inputline`, it stores the integer in the location pointed to by `value` and the location of the character immediately after the last digit at the location pointed to by `nextchar`, and returns 1. However:
   i. If the integer is too large (positive — see the second hint), it reads all the digits, and returns 2; the value stored in the location pointed to by `value` need not be correct.
   ii. If the integer is too small (negative — see the second hint), it reads all the digits, and returns 3; the value stored in the location pointed to by `value` need not be correct.
(c) Otherwise, *`getint`* does not change the values stored at the location pointed to by `value`, but it stores the value of `inputline` in the location that `nextchar` points to, and returns 0.

Your function is to be in a file called "getint.c".

*Hint:* Think back to the "if argument *n* is invalid" comment in assignment 2 problem 6. In a similar way, the arguments passed into this function could be invalid. Use what you have learned in lectures 9–14 about pointers, arrays, strings, and buffers, to think about how these arguments could be given invalid values, and how to detect those invalid values.

*Hint:* The maximum positive integer is represented by the macro **`INT_MAX`**, and the minimum negative integer is represented by the macro **`INT_MIN`**, both of which are defined in the include file "limits.h".

*Hint:* There is a file that provides an interface (a *`main`* routine) available on SmartSite; it is called "getint-driver.c" and is on the CSIF at "`/home/bishop/ecs30/getintdriver.c`". If you download it and compile it with your function (see **Compiling and Executing Your Program** for details on how to do this, but use the file names in this problem), you can focus on writing the function. Also, we have supplied the input files that Gradebot uses. These are available on the CSIF as "`/home/bishop/ecs30/hw3-6-input`/*n*", where *n* is a number between 1 and 10 inclusive.

## Debugging

You ***must*** put your *function* in a file called "calc.c". ***If you call your file anything else, or submit something other than a C program (for example, a text or PDF file containing the code), we will not grade it and you will get 0!***

7. (*15 points*) The program *calc.c* discussed in class has another bug. At the "operand" prompt, type any non-integer. The program will go into an infinite loop. Find out why and fix it. Place your explanation for the error, and how you fixed it, in the header comment.