Homework 1

Due: October 16, 2015 (**Note extension**) **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. (*5 points*) What option should you give to *rm*(1) to make it ask you verify that each file is to be deleted?
2. (*5 points*) Is it possible to restore a file that has been removed?
3. (*5 points*) What command was run to produce the process with PID 1?
4. (*5 points*) What two subdirectories of /usr/local have names beginning with "Fortify"?

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.

5. (*42 points*) In MyProgrammingLab, please do the first question in each subsection of the following sections: 2.1–2.3, 2.5, 4.2–4.4, 4.6–4.8, 5.2–5.4, 5.6–5.8 (except exercise 10391). 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.
6. (*42 points*) In the textbook, please do questions 2, 3, and 7 in the Review Questions section of chapter 2, questions 3 and 4 in the Review Questions section of chapter 4, and questions 5 and 6 in the Review Questions section of chapter 5.

Programming

7. (*90 points*) Counting from one number to another is a common function of computers. Your program is to accept 3 inputs: a starting number, an ending number, and an increment. All of these are integers. Your program is to print the numbers from the starting number to the ending number inclusive, counting by the increment. So for example, if the starting point is 5, the ending point is 30, and the increment is 5, your program is to print:

   ```
   5 10 15 20 25 30
   ```

   Do not print any numbers beyond the ending point!

   The interface of your program must look *exactly* like this (your input is in bold; what the computer types is in normal font):

   ```
   Starting point: 5
   Ending point: 30
   Increment: 5
   5 10 15 20 25 30
   ```

   No extra spaces after the last number, or blank lines, may occur. As another example:

   ```
   Starting point: 5
   Ending point: 12
   Increment: 2
   5 7 9 11
   ```

There is to be one space between each number, and *no* space (or blank lines) trailing the final number in the output.

Errors arise when the starting point is less than the ending point and the increment is non-positive, or when the starting point is greater than the ending point and the increment is non-negative. In these cases, the error messages that you output must look *exactly* like this:

```
Increment must be > 0 if begin < end
```

and

```
Increment must be < 0 if begin > end
```

They also must be written on the standard error. To do so, replace the `printf` you would normally write with the appropriate one of these:

```
fprintf(stderr, "Increment must be > 0 if begin < end\n");
```

and

```
fprintf(stderr, "Increment must be < 0 if begin > end\n");
```

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

Debugging

8. (*48 points*) The program *vis.c* reads characters from the standard input and prints them on the standard output after expanding any non-printing characters to their C character escape sequence.

   The relevant characters, and the C escape sequences to be printed when those characters are encountered, are:

character	*print as*	*character*	*print as*
newline (^J)	\n	horizontal tab (^I)	\t
vertical tab (^K)	\v	backspace (^H)	\b
carriage return (^M)	\r	form feed (^L)	\f
bell (^G)	\a	NUL (^@)	\0
backslash	\\	anything else	\x*xx*

   The "anything else" entry means that any non-printing character other than the ones named in the table is to be printed as a sequence of two hexadecimal digits preceded by a backslash and a lower-case letter "x". When the escape sequence for a newline is printed, the program is to skip to the next line.

   Unfortunately, the program as saved in *vis.c* will not even compile, let alone run. And the programmer thoughtlessly left off all the comments. Hence, your mission: comment the program, and fix it so it works as described above! You are to turn in a corrected source program, with comments describing the changes you made to get it to work.

   As an example of how to do this, let's say the program contains the following:

```
x = 8;
...
if (x = 9)
    printf("x is 9\n");
else
    printf("x is %d\n", x);
```

   The error is clearly in the condition of the **`if`** statement, as that assigns 9 to `x` (and does not compare 9 to `x`), and so always evaluates to true. Here's the sort of comment we want:

```
x = 8;
...
/*
 * original line was:
 *    x = 9
 * changed from an assignment to a test
 * because the if statement meant to test
 * the value of x, not change it
 */
if (x == 9)
    printf("x is 9\n");
else
    printf("x is %d\n", x);
```

Note the comment says what the change was, and why it was made.