Programming Assignment #1 **Due: 2/15**

1. [100 points] Write a C program *finddups* that will scan a set of files looking for duplicates it will organize into groups. The files are specified on the command line:

   `$ ./finddups` *`fileOrDirectory*`*

   In addition, any directories specified on the command line are traversed recursively, looking for more regular files (and directories) to compare, all the way down the directory tree. If there are no command line arguments, the program acts on the current directory. (i.e., "`$ ./finddups`" does the the same thing as "`$ ./finddups .`").

   finddups will print the contents of all groups of two or more identical files to standard output. For each group with n members, the format will be:

   `n 1 pathOfFirstFile`
   `n 2 pathOfSecondFile`
   `n 3 pathOfThirdFile`
   `...`
   `n n pathOfNthFile`

   For example, suppose there are three identical files `moe`, `larry`, and `curly` and one additional file, `shemp`, that differs. `curly` and `shemp` reside in the current directory, while `moe` and `larry` reside in a subdirectory of the current directory named `brothers`.

   The command

   ```
   $ ./finddups brothers curly shemp
   ```

   will output a single group of the three identical files:

   ```
   3 1 brothers/moe
   3 2 brothers/larry
   3 3 curly
   ```

   (The order of the files within the group is not important.)

   Additional spaces, newlines, or other output will cause points to be deducted.

   All path information will be preserved in the output: Relative file names will remain relative. Absolute file names will remain absolute.

   Ignore symbolic links, devices, and all other non-regular files (cf. *stat(2)*). (Directories, of course, you will recur on.)

   If you find that a file might be a duplicate and you are unable to read it, print an error message on standard error, but do not stop examining other files. Also print an error message (but again, don't stop) if you find a directory that you cannot traverse.

This program should be usable on whole hierarchies: "`finddups ~`" or even "`finddups /`" (if you ignore errors) are not outside the realm of possibility[1] so efficiency is a concern.

The theoretical efficiency of this program is $O(N^2M)$ where N is the total number of files being examined and M is the mean length of a file in bytes. Your program should do all it can to minimize execution time. In particular, it should in general *not* have to compare every file, byte-by-byte, to every other file. A hint on a possible way to go about this: `stat(2)`.

[1] Please use discretion: Other people might be logged in!