ECS 40 **Program #6 (50 pts, 2.5 hours)** **Winter 2016**

Due: Wednesday, February 24th, 11:59 PM. Executable Name: FileFind.out

Filenames (case sensitive): authors.csv, directory.cpp, directory.h, list.cpp, list.h, main.cpp, and Makefile.
New Concepts: templates, inheritance, and double linked lists.

You are to write a program that reads a file that recursively lists the files in a directory. There are four classes: ListNode, List, File, and Directory. The Directory class is derived from the File class. To allow for polymorphism, the List declared in main() will hold File*, not File. As the program reads the file it will use the first letter to determine whether it should create a new Directory or new File. It then can insert into the list the pointer. For find, it will need to create a File based on the user input, and then pass the address of the File to List::find(). There is a simple way to repeatedly create such a File inside a loop without calling new. See if you can figure out how I did it.

For this assignment, I am not going to give you step by step instructions for the development of the program. It is time for you to develop a program on your own. I suggest that you develop from top down using stubs.

1. main() calls a read function, and then enters a loop. main() prints the "Not found" if list.find() returns false.
2. ListNode class is a template class with both a prev and next ListNode<T> pointers. Because the T data stored is a pointer to an object that is created using new, it will be up to the destructor of ListNode to delete it.
3. List class is a template class with ListNode<T> *head, and *curr, that is sorted based on the overloaded comparison operators of the T class that compare the names of the Files.
   3.1. To get things up and running, initially I would just use the List as a single linked list. However, it is worth five points of the fifty to have curr move forward and backward within the list.
   3.2. Since the list knows that it will always be storing pointers as the data in the ListNodes, it will usually dereference the data when using it. Be careful when comparing dereferenced T's that the one in the list is always on the right side of the comparison! This permits the operator of the proper class, File or Directory, to be called.
   3.3. There is five points extra credit if you write both the find(), and insert() without using any additional ListNode<T> pointers, e.g. the ptr and prev used in lecture.
   3.4. The methods are: a constructor, a destructor, bool find(), void insert(), and void print().
4. File class has only a protected char *name for data that must be dynamically allocated. The code for each of the methods is no more than three lines long!
   4.1. The non-virtual methods are all public: a constructor, and const char* getName().
   4.2. The virtual methods are: a destructor, bool find(), void insert(), operator==, operator<, and operator>.
      4.2.1. The find() method just prints the name of the file.
      4.2.2. The insert() method does nothing!
      4.2.3. The comparison operators just use strcmp().
5. Directory class is publicly derived from File, and also contains a List<File*>, named files. Thus, main's List is actually a List of Lists!
   5.1. The methods it has are: a constructor, a destructor, and versions of all the virtual methods of File.
      5.1.1. find() prints the List of the Directory if it's name is a <u>perfect</u> match, otherwise it calls the find of its List.
      5.1.2. insert() does nothing if its name is a <u>perfect</u> match, otherwise it calls the insert of its List.
      5.1.3. The comparison operators are used by List, and use strncmp() to compare the name of the directory with the beginning of the name of the data being compared. They are a little tricky. For example, "40/p4 == 40/p4/private" is true because the p4 is followed by a '/' in the second operand, but "./handin/p4/davis == ./handin/p4/davistemp" is false because the "davis" in the second operand is neither the end of the string, nor followed by a '/'.

Further specifications:
1. const must be used wherever possible in function headings. This includes parameters, return types, and functions themselves. Note that there is no need to label parameters and return types that are passed by value as const.
2. You may assume that all input will be valid, and not require any form of range checking.
3. Your Makefile file must use g++ with the –g –Wall –ansi options for compiling and linking. You will lose one point for each warning. Remember to omit any mention of list.cpp and list.o in your Makefile.
4. Handin to p6 in cs40a
5. You will my executable and some source files in ~ssdavis/40/p6.

```
[ssdavis@lect1 p6]$ find . -exec ls -ld {} \; > p6Files.txt
[ssdavis@lect1 p6]$ cat p6Files.txt
drwx------ 3 ssdavis users 4096 Feb 16 16:40 .
drwx------ 2 ssdavis users 4096 Feb 16 16:40 ./private
-rw-r--r-- 1 ssdavis users 1184 Feb 16 16:36 ./private/directory.cpp
-rw-r--r-- 1 ssdavis users 752 Feb 16 16:33 ./private/file.cpp
-rw-r--r-- 1 ssdavis users 2279 Feb 16 16:33 ./private/list.cpp
-rw-r--r-- 1 ssdavis users 1033 Feb 16 16:37 ./private/main.cpp
-rw-r--r-- 1 ssdavis users 354066 Feb 16 16:33 ./private/cs40aFiles
-rw-r--r-- 1 ssdavis users 396 Feb 16 16:33 ./private/directory.h
-rw-r--r-- 1 ssdavis users 399 Feb 16 16:33 ./private/file.h
-rw-r--r-- 1 ssdavis users 475 Feb 16 16:33 ./private/list.h
-rw-r--r-- 1 ssdavis users 393 Feb 16 16:37 ./private/Makefile
-rwxr-xr-x 1 ssdavis users 20776 Feb 16 16:37 ./private/FileFind.out
-rw-r--r-- 1 ssdavis users 750 Feb 16 16:40 ./p6Files.txt
-rwxr-xr-x 1 ssdavis users 20776 Feb 16 16:37 ./FileFind.out
-rw-r--r-- 1 ssdavis users 354066 Feb 16 16:33 ./cs40aFiles.txt
-rw-r--r-- 1 ssdavis users 29691 Feb 16 16:33 ./40files.txt
[ssdavis@lect1 p6]$
[ssdavis@lect1 p6]$ FileFind.out p6Files.txt
Please enter a file name (Done = exit): .
./40files.txt
./FileFind.out
./cs40aFiles.txt
./p6Files.txt
./private
Please enter a file name (Done = exit): ./private
./private/FileFind.out
./private/Makefile
./private/cs40aFiles
./private/directory.cpp
./private/directory.h
./private/file.cpp
./private/file.h
./private/list.cpp
./private/list.h
./private/main.cpp
Please enter a file name (Done = exit): ./private/Makefile
./private/Makefile
Please enter a file name (Done = exit): ./private/makefile
Not found.
Please enter a file name (Done = exit): Done
[ssdavis@lect1 p6]$
 [ssdavis@lect1 p6]$ FileFind.out 40files.txt
Please enter a file name (Done = exit): 40
40/notes
40/p2
40/p3
40/p4
40/p5
40/p6
40/tests
Please enter a file name (Done = exit): 40/p6
40/p6/FileFind.out
40/p6/cs40aFiles
40/p6/p6Files
40/p6/private
Please enter a file name (Done = exit): 40/p5/main.cpp
40/p5/main.cpp
Please enter a file name (Done = exit): 40/p5/Main.cpp
Not found.
Please enter a file name (Done = exit): Done
[ssdavis@lect1 p6]$
```