**CISP1020 Lab2, Linked List as a Queue and Binary Files**

**General Description**

This application implements the buying and selling of stocks.

//Written for Linux/Unix environment in C language

**Relevance**

Stocks are bought and sold every second, every day. Individual investors can buy/sell stocks on web sites such as Scottrade and Ameritrade. Mutual funds are made up of stocks. Most employee retirement accounts contain many mutual funds (made up of stocks). Economic indices that you hear about on the news such as the S&P 500 (made up of the top 500 large capital companies in the United States) are used to report the performance of large companies in the US.

When stocks are sold, the oldest stocks are usually the ones sold first in the hopes that they are more than one year old which lowers the amount of taxes the investor has to pay. Gains made from selling a stock that is more than one year old are called long term gains. The tax rate for long term gains is 0% for those in a 10% to 15% income tax brackets, 15% for those in the 25% to 35% tax brackets, and 20% for the wealthiest citizens who fall into the 39.6% income tax bracket. Short term gains, taxed as ordinary income at your normal tax rate of 10 – 39.6%, are stocks that have been held for less than one year (<https://www.investopedia.com/articles/personal-finance/101515/comparing-longterm-vs-shortterm-capital-gain-tax-rates.asp> 9/11/2018).

Programming is always inter-disciplinary. We create programs for people who want to use them in a lot of different contexts, for example, stocks. So, we, as programmers, end up having to learn enough about a subject to create an application for those people. The code we’re writing could be used as part of a brokerage website like E\*TRADE® for clients to buy and sell stocks.

**Detailed Description**

The application begins by printing a welcome message.

Welcome to YourTrade.com

Then a main menu of choices for the user:

Reporting, buying or selling?

(0=quit, 1=report, 2=buy, 3=sell):

Print a goodbye message when the user quits such as:

Thank you for trading with YourTrade.com

Use a queue of stock\_t structures. A stock\_t structure looks like:

#define MAX\_TICKER\_LENGTH 6

typedef struct stock\_t {

char ticker[MAX\_TICKER\_LENGTH];

date\_t date; // date bought

int numShares;

double pricePerShare;

} stock\_t;

A date\_t structure looks like:

typedef struct date\_t {

int month, day, year;

} date\_t;

**Create the following files in a directory named c1020axxLab2 where xx is your user ID number.**

* date.h: contains the above date\_t structure
* stock.h and stock.c: contain the stock structure and any constants and stock function prototypes
* linkedList.h and linkedList.c: contain the list structure and any constants and list function prototypes and functions. Use the files from the pair programming. This lab will use the linked list as a queue.
* main.c contains the main function and, possibly, other functions such as functions to report, buy and sell.
* Makefile: a makefile to make each source file separately or the entire program using the gcc compiler (not g++) and to “clean” the directory (remove all object files and the executable)
* Reporting: Print all stocks owned to standard output. After printing stocks owned, ask the user which stock to report on and print all of the details of that stock. Stocks owned are stored in binary files with the stock ticker symbol as the name in all capital letters. For example, Apple stocks would be in a file called AAPL.bin and Google stocks in a file called GOOG.bin. In order to report, the program should use the following types and functions from dirent.h (“directory entry” header file). A single directory entry has information in it such as file name.
  + DIR\* dirPtr: a pointer to a DIR, a directory. Used by the opendir function to open a directory for reading. This is analogous to a FILE\* (file pointer) used for reading/writing from/to files that are not directories.
  + struct dirent\* dirEntry: a pointer to a directory entry structure. The readdir function returns one of these, a directory entry, which can be used to get the name of the file that the entry pertains to, dirEntry->d\_name
  + DIR\* opendir( char\* ): takes the path of a directory such as “.” for the current directory and opens it. It returns a DIR\* or NULL if the open was unsuccessful (i.e., the directory doesn’t exist). The opendir function is analogous to the fopen function used for opening files.

dirPtr =opendir( “.” );

* + struct dirent\* readdir( DIR\* ): takes a DIR\* (a directory) and returns a structure representing the next entry in the directory. The function returns NULL if there are no more entries in the directory.

while( (dirEntry = readdir( dirPtr )) != NULL ) {

// use dirEntry->d\_name and see if it’s a \*.bin”

// file. If it is, make a string out of the name

// only (without the “.bin”), and print it

}

* + int closedir( DIR\* ): closes a directory pointer just like you would close a file after reading from it.

closedir( dirPtr );

* Sample execution of reporting:

Stocks Owned

------------

AAPL 73

GOOG 56

INTC 10

Enter stock ticker symbol: AAPL

Ticker Purchase Date Shares Price Per Share

---------------------------------------------

AAPL 2/18/1981 10 $ 27.25

AAPL 9/ 2/1983 7 $ 38.00

AAPL 6/ 3/1986 15 $ 37.87

AAPL 12/31/1990 5 $ 43.00

AAPL 12/29/2000 26 $ 14.88

AAPL 10/ 1/2010 5 $ 282.52

AAPL 9/ 8/2015 5 $ 111.69

If you’re dying for more information, see the links below, but the above information is enough for your lab. <http://pubs.opengroup.org/onlinepubs/007908775/xsh/dirent.h.html> or <https://en.wikibooks.org/wiki/C_Programming/POSIX_Reference/dirent.h>

* Buying: Prompt the user for the ticker symbol, the number of shares to buy and the price per share. Open a binary file for that ticker symbol in append mode as shown below:

FILE\* output;

output = fopen( filename, "a" );

Then, get the date from the system, fill up a stock\_t structure with this information and write the structure to the binary file. Close the binary file. For more information on open modes, see <http://www.cplusplus.com/reference/cstdio/fopen/> although the above is enough for your lab.

* Selling: Prompt the user for ticker symbol of stock then open that stock’s binary file for reading. If the file doesn’t exist, prints a message indicating that the user doesn’t own any of that stock. Otherwise, read all of the stocks from the file and shift them onto the linked list of stock\_t structures (the list is treated as a queue) then close the file. Print out how many shares of that stock the user has. Ask the user for the number of stocks to sell and the current stock price. If the user doesn’t own that many shares total, output an error message and reprint the main menu above (report, buy, sell or quit). Otherwise, go through the list as a queue of stocks removing all nodes necessary for the number of shares needed and calculating and printing the total price to buy the stocks, the total selling price, and the gains (or losses). Open the file again in write mode and write the entire file from the updated contents of the list from beginning to end (as a queue). Close the file. If user sells all shares of a stock in a file, then the program has to delete the file with the remove function in stdio.h which takes a filename (char \*) as a parameter.

remove( filename );

Assume:

* User input will be of the correct format. For example, if you ask for an integer you will get a valid integer.
* Max price per share < 1000
* Max ticker symbol length = 5 chars
* Stocks are listed in the file in the order they were purchased

See my binary read and write programs for examples of reading and writing stocks from/to binary files. Use sample binary files: AAPL.bin, INTC.bin and GOOG.bin to begin testing your program. All of these are in a zip file in our online Course Content.

**Due Date (see D2L)**

**Code:** You do not need to turn in a printout. Copy your c1020axxLab2 folder to my turnin directory. For example (***substituting your username for c1020axx***):

cp -r c1020axxLab2 /export/home/students/Arnold\_turnin/