Lab 2, Linked Lists, Binary Files

# General Information

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 and 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. If you’re interested, read more at [Investopedia.com](http://www.investopedia.com/articles/personal-finance/101515/comparing-longterm-vs-shortterm-capital-gain-tax-rates.asp)

# Detailed Information

This application implements the buying and selling of stocks. It begins by printing a welcome message

Welcome to YourTrade.com

Then a main menu of choices for the user is output.

Reporting, buying or selling?

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

The program ends with a goodbye message.

Thank you for trading with YourTrade.com

Use a doubly linked list 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. Example: a print stock function.
* node.h, node.c: contain at least the node typdef and initNode function
* list.h and list.c: contain the doubly linked list structure and any constants and list function prototypes. Start with the pair programming list files and add function(s) for selling (e.g., searching the list). All functions that operate on the list as a whole should go here.
* 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: Prints 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 );

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: Prompts the user for the ticker symbol, the number of shares to buy and the price per share. It opens a binary file for that ticker symbol in append mode:

FILE\* output;

output = fopen( filename, "a" );

gets the date from the system, fills up a stock\_t structure with this information and writes the structure to the binary file. Closes 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: Prompts the user for ticker symbol of stock. Opens 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, the program reads all of the stocks from the file and puts them in a queue then closes the file. The program prints out how many shares of that stock the user has and asks 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 queue of stocks removing 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 writes the entire file from the updated contents of the list. Close the file. If the 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 );

Assumptions

* User input will be of the correct format. For example, if you ask for an integer you will get a valid integer although it may not be in a valid range.
* Max price per share < 1000
* Max ticker symbol length = 5 chars (so a C-string needs 6 characters, an extra for the null character)
* Stocks are listed in the file in the order by date that they were purchased

See the 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 in a zip file that is one of the assignment files.

# Sample execution

In sample execution, *mm/dd/yyyy*, means the current date.

Welcome to YourTrade.com

Reporting, buying or selling?

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

Stocks Owned

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

AAPL 15

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/ 8/2015 5 $ 111.69

Reporting, buying or selling?

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

Enter stock ticker symbol: AAPL

Enter number of shares: 10

Enter stock price: 147.84

10 shares of AAPL stock purchased at $147.84 per share.

Reporting, buying or selling?

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

Stocks Owned

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

AAPL 25

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/ 8/2015 5 $ 111.69

AAPL *mm/dd/yyyy* 10 $ 147.84

Reporting, buying or selling?

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

Enter stock ticker symbol: DISCA

Enter number of shares: 5

Enter stock price: 30.93

5 shares of DISCA stock purchased at $30.93 per share.

Reporting, buying or selling?

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

Stocks Owned

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

AAPL 25

DISCA 5

GOOG 56

INTC 10

Enter stock ticker symbol: DISCA

Ticker Purchase Date Shares Price Per Share

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

DISCA 7/15/2021 5 $ 30.93

Reporting, buying or selling?

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

Enter stock ticker symbol: AAPL

You own 25 AAPL shares.

Enter number of shares: 10

Enter stock price: 147.84

Shares sold: $1478.40

Shares bought: $272.50

Gain: $1205.9

Reporting, buying or selling?

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

Enter stock ticker symbol: AAPL

You own 15 AAPL shares.

Enter number of shares: 15

Enter stock price: 100

Shares sold: $1500

Shares bought: $2036.85

Loss: ($536.85)

Reporting, buying or selling?

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

Stocks Owned

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

DISCA 5

GOOG 56

INTC 10

Enter stock ticker symbol: AAPL

You do not own any AAPL stock.

Reporting, buying or selling?

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

Thank you for trading with YourTrade.com

Relevance

1. Our program uses a doubly linked list as a queue. List another context (not stocks, but something else) where data is manipulated first-in-first-out.
2. A double linked list can also be used as a queue or a stack. Why isn’t a doubly linked list useful for data that needs to be accessed in a random or (not front to back or back to front)?

Rubric

For any credit on this lab, the program must have all of the C source code files and header files listed above, read data from a binary file into a doubly-linked list with one stock\_t structure in each node, and use this list when selling stocks.

* (10 points) Comments and style
* (4 points) Relevance questions
* Program correctness (86 points)
  + (5 points) Makefile
  + (2 points) Non-existent file
  + (2 points) Empty file
  + (10 points) Reporting
  + (10 points) Buying
  + (10 points) Reading from the binary file and inserting into the list
  + (5 points) Selling resulting in decrementing one node’s stock shares
  + (5 points) Selling resulting in removing exactly one node
  + (5 points) Selling resulting in removing more than one node and decrementing the shares in a node
  + (5 points) Selling resulting in removing all nodes from the list
  + (10 points) Saving a binary file
  + (5 points) No memory leak
  + (12 points) Miscellaneous

# What to Turn In

1. In the file comment header of main.c, write the answers to the relevance questions. **Don’t forget to write answers to the relevance questions**.
2. Copy the entire directory you created, c1020axxLab2 (substituting your number for the xx) that has the lab files in it to the instructor’s turnin directory on the Linux server.