PROGRAMMING FINANCE: FINAL PROJECT

ACCOUNT MANAGEMENT SYSTEM: DESCRIPTION

Text Files Used:

- 5 sample result files named *Result_1.txt*, *Result_2.txt*, *Result_3.txt*, *Result_4.txt*, and *Result_5.txt*. Each of the these text files contain all the company symbols, prices per share, and dates.
- *current_portfolio.txt* contains information about the stock account like the company symbol, number of shares, price per share, and total price.
- *portfolio_history.txt* contains the total portfolio value, cash balance, date and time. This file has been used to plot a graph of the portfolio value over a period of time.
- *stock_transaction_history.txt* contains the buy and sell transaction history of the user.
- *bank_trasaction_history.txt* contains the deposit and withdraw history in the bank account.
- *portfolioValue_cashBalance.txt* contains the latest total portfolio value and the cash balance, both of which keep changing.

Functions Implemented:

- Class 'Account'
  - read_cash_balance(): To read the latest values of cash balance and total portfolio value from *portfolioValue_cashBalance.txt*.
  - write_cash_balance(): To write the latest values of cash balance and total portfolio value into *portfolioValue_cashBalance.txt*.
  - put_graph_values(): To update *portfolio_history.txt* so as to plot a graph of the portfolio value.
  - write_bank_transaction_history(): To update *bank_trasaction_history.txt* which will contain all bank transaction history like how much money was deposited/withdrawn.
- Class 'StockAccount'
  - StockAccount(): Constructor that initializes the head and tail of each Node to NULL.
  - void bSort(Results): To keep the doubly linked list in sorted order after each buy/sell operation (Bubble sort).
  - void stock_menu(Results, Results, Results, Results, Results): To display the options available to the user. When the user selects an option, a particular function that implements that option will be called using switch case.
  - Results random_file_selector(Results, Results, Results, Results, Results): To keep selecting a different Result.txt file each time it is called so that the values are not constant.

- ➢ void display_price(Results): To display the price of a particular stock.
- ➢ void add_node_to_DLL(Node *): To add a new node to the double linked list.
- ➢ void buy_stock(Results): To enable user to buy stocks.
- ➢ void sell_stock(Results): To enable user to sell stocks.
- ➢ void from_portfolio_to_DLL(): To populate the doubly linked list with information from the *current_portfolio.txt* file when the program starts.
- ➢ void write_current_portfolio(Results): To write into *current_portfolio.txt* after each buy/sell operation.
- ➢ void read_current_portfolio(): To print the information in *current_portfolio.txt* whenever the user wants to.
- ➢ int search_DLL(Node *): To search the entire doubly linked list and ensure that there's only one copy of each symbol in the list.
- ➢ void write_transaction_history(string, string, int, double, double): To write all the stock buy/sell transaction history into *stock_transaction_history.txt.*
- ➢ void read_transaction_history(): To print stock buy/sell transaction history from *stock_transaction_history.txt.*
- ➢ void plot_graph(): To plot graph using MATLAB.

❖ Class 'BankAccount'

- ➢ void bank_menu(): To display the options available to the user. When the user selects an option, a particular function that implements that option will be called using switch case.
- ➢ void view_balance(): To display the current cash balance in the bank account.
- ➢ void deposit_money(): To enable the user to deposit money into his/her account.
- ➢ void withdraw_money(): To enable the user to withdraw money from his/her account.
- ➢ void read_Btransaction_history(): To print the history of all transactions that have taken place.

❖ Class 'Results'

- ➢ void get_data(string): To copy the information in Results.txt files to a data structure (Results).

❖ Class 'Node'

- ➢ Node(): Constructor to initialize the *next* of each Node to NULL.

Data Structure Used: Doubly linked list in which each node stores the company symbol and the number of shares.

Design Pattern Used: Structural Design Pattern (Bridge) to display time when the user exits the program.