Adam Castillo

Mai Xiong

CSCI 156 Fall 2017

MW 3:30-4:45 PM

Automated Stock Trading Socket Program

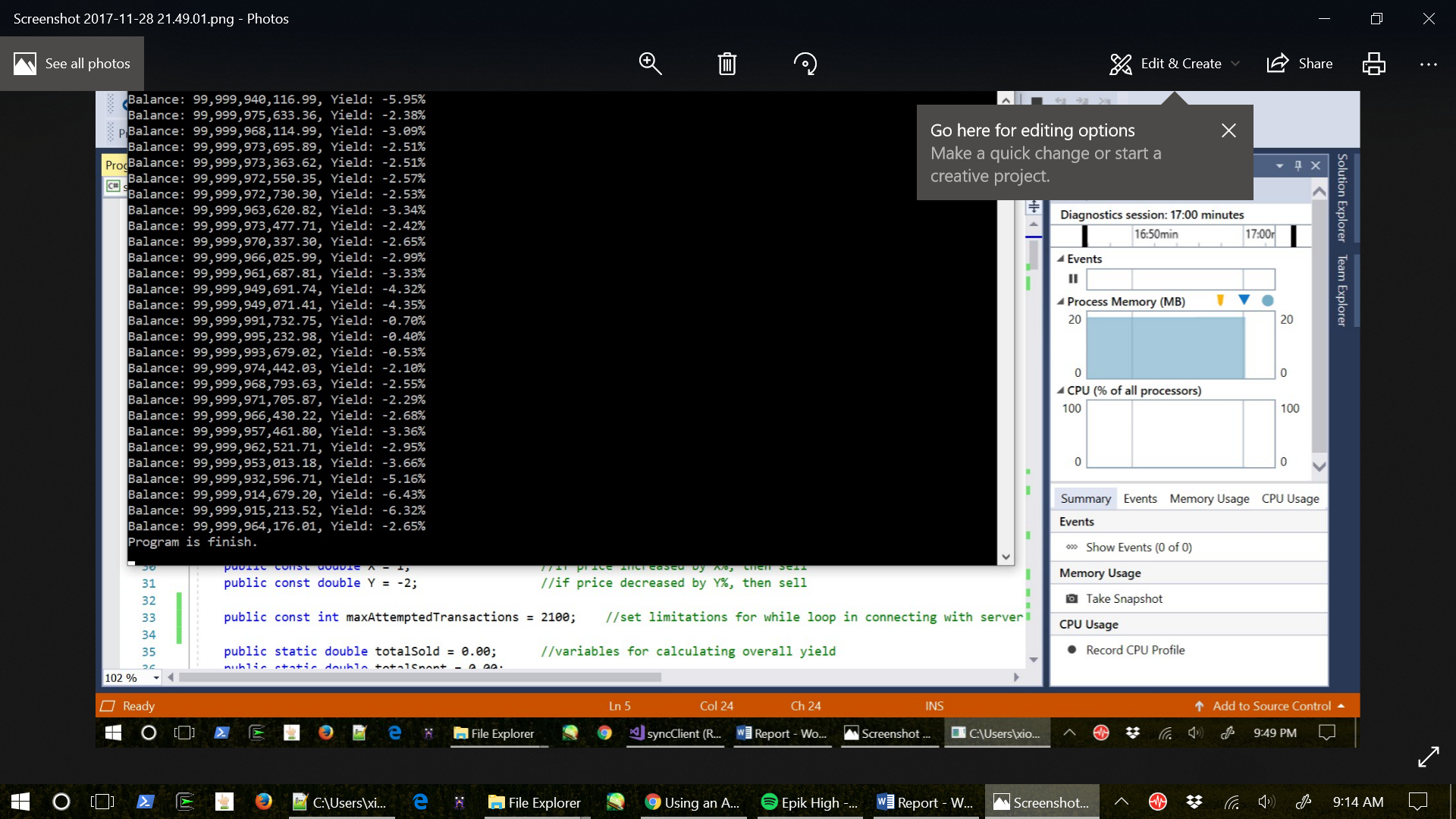
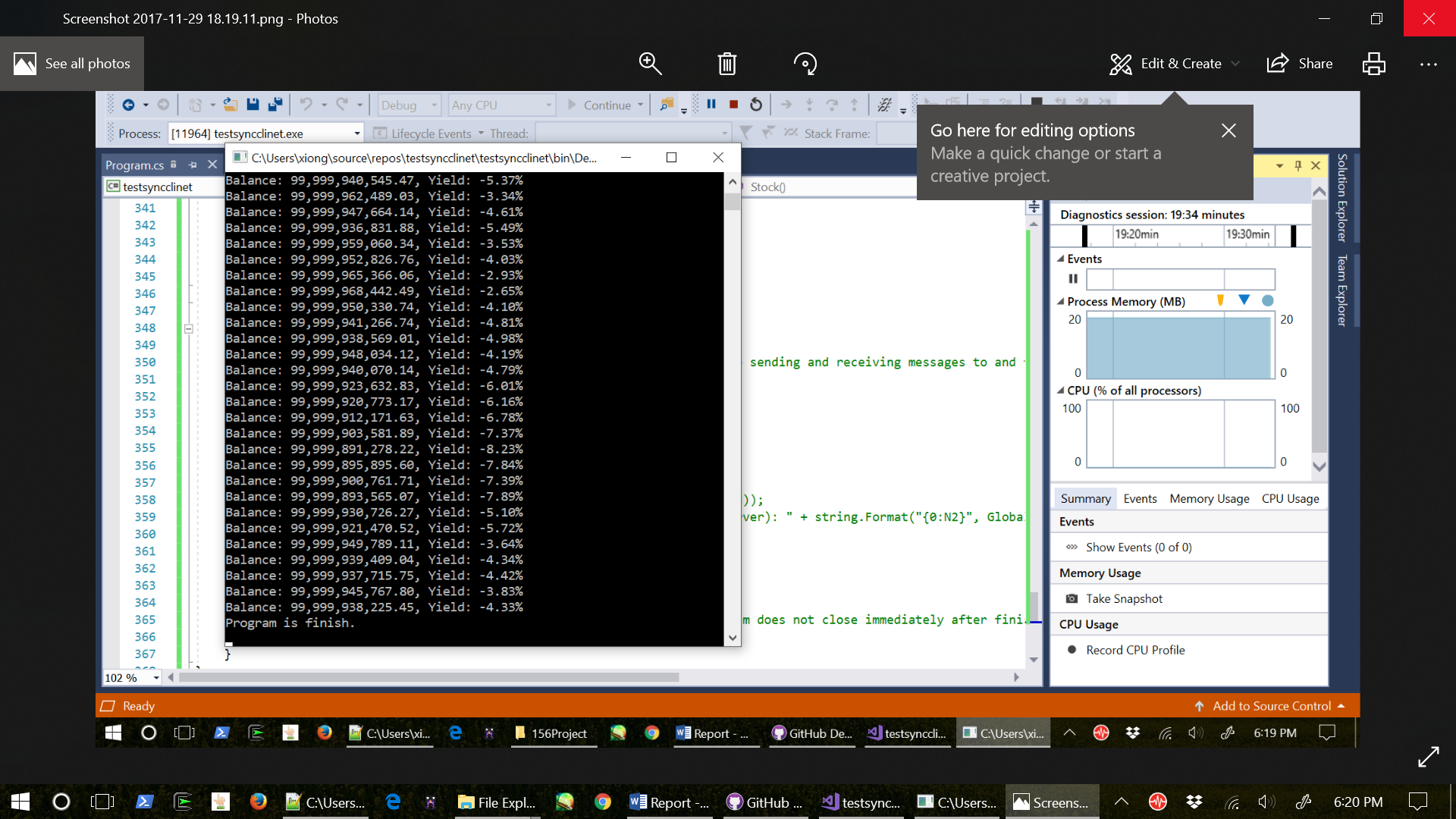
C# has two ways of using the Socket class: asynchronous or synchronous. They use the same Socket class, but different methods. The programs use TCP protocol to ensure successful delivery of the message. The random port 17000 is use. The client only uses one socket to connect to the server. In this program, asynchronous methods (starting with “Begin”) are used to implement the server, and synchronous methods are used to implement the client. Asynchronous use of the sockets allows the program to continue without waiting for network operations to be completed or to continue processing other network service requests. The server creates a new thread to complete the asynchronous processes. Each time BeginAccept(), BeginReceive(), or BeginSend() is executed, a new thread is created to execute the corresponding action. Synchronous use of sockets does not allow the program to continue to the next step until the network operation is completed or until a connection request is received. In this program, the server uses asynchronous methods to manage the sockets, so it can continue to accept requests from other clients, and at the same time completing the previous requests. The client uses synchronous methods on its single socket since it only needs to communicate to the server.

The *Synchronous Client Socket Example* and *Asynchronous Server Socket Example* on Microsoft’s website were used as the base of our codes. The *Synchronous Client Socket Example* connects to the server, sends and receives data, and then it closes the socket. The *Asynchronous Server Socket Example* listens for connections, and for each connection, it receives and sends data, and then it closes the socket. We modified the examples to fit the stock trading programs and to send and receive more than one message. We manually created a thread in the client program to report the client’s balance and overall yield every 10 seconds.

First, the server program must start running before the client(s) program start. Here is an overview of the process taken by the server and the client(s):

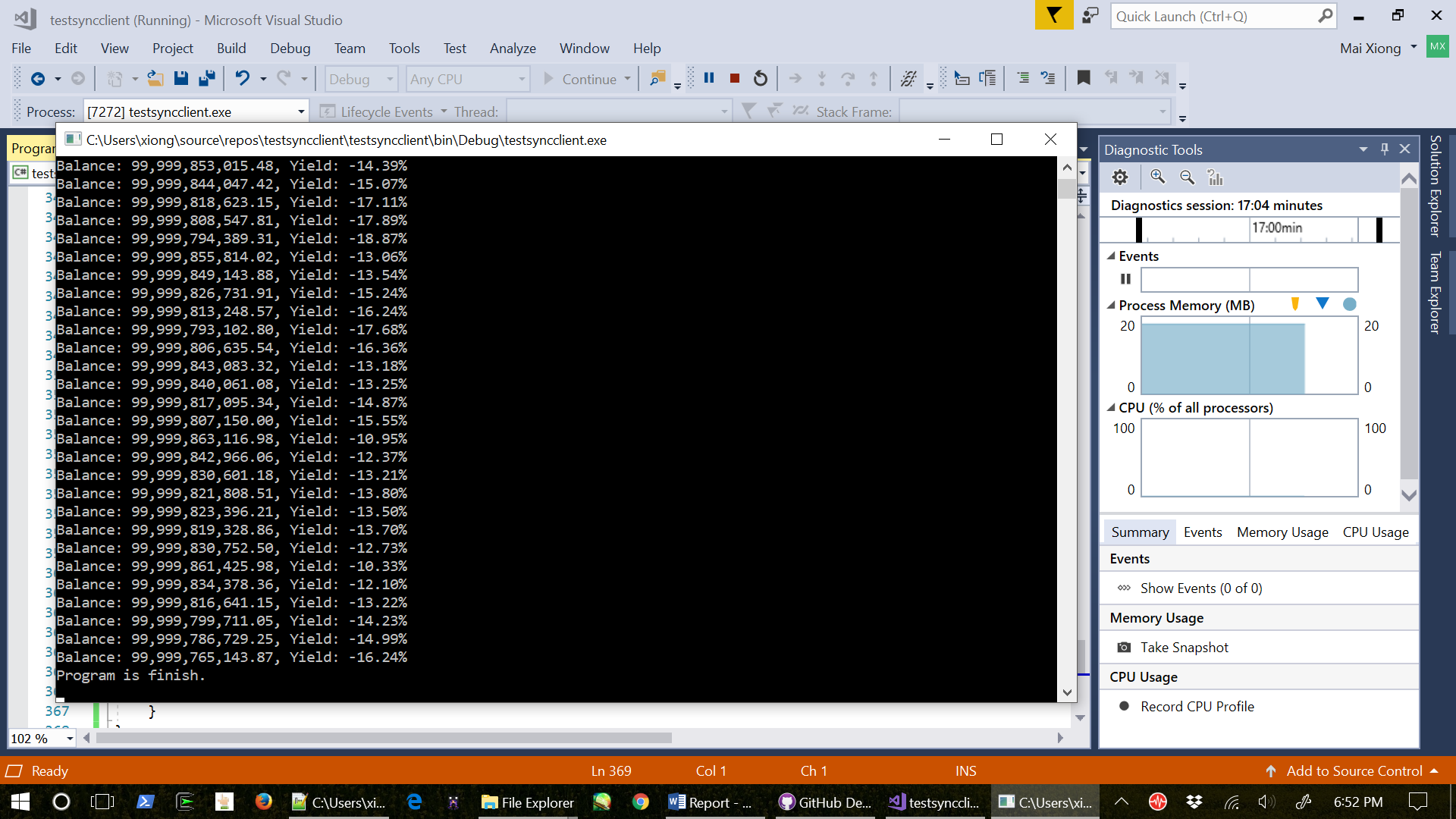
|  |  |  |
| --- | --- | --- |
| Server Program | The asynchronous part of Server Program\* | Client Program |
| 1. Read Data.txt 2. Create Dictionary *stockInfo* for holding the Stocks and their information. 3. Listen for connection requests, and if there is a request for connection, create connection with Client (\*asynchronous part of the server in dealing with requests from a certain client) 4. Repeat step 3 (to stop server, close the server program). | 1. Get the socket that handles the client request. 2. If the socket’s IP address and port number matches a previous address, add a new balance into dictionary balances. 3. Begin receiving from Client. 4. If the received message from the client is  * “Give me the Stock List<EOF>” then send the stock names and their current prices to the client. * “BALANCE<EOF>” then send the balance of the client to the client * [Transaction message] then check if the message is a BUY or a SELL. Update Stock information if transaction is successful and send the client the message, “Success<EOF>” else send “Failure<EOF>”. Or send  “10PercentFAIL<EOF>” if the transaction failed by 10% rate. * Does not have “<EOF>” in message then continue receiving the message.  1. If client closed their socket, then server will also close the socket used for that client. 2. Repeat steps 6-8 until client close socket. | 1. Requests connection to server 2. Confirmed connection is complete 3. Create and Start thread for reporting balance and yield every 10 seconds. 4. Send message to server to give the stock list. 5. Receive message of stock list and their current prices (example: APPL 20.00 MSFT 21.22). 6. Initialize Dictionary *StockList* if is the first time requesting for the stock list, else only update current price of the Stocks. 7. Create transaction message by deciding whether to sell or buy. 8. If transaction message is blank, then client decided not to buy or sell else send transaction message to Server. 9. Receive message from Server indicating whether the transaction was a success or not. If success, update yield, and Stock information. 10. Client sleeps for 1 second if transaction is a success. 11. Repeat steps 4-10 until 2100 *maxAttemptedTransaction* is reached. 12. Close socket. 13. Stop reporting thread. 14. Client program finishes. |

The stocks and the stocks’ prices were downloaded from Yahoo Finances. When the client buys, it buys a random amount (from 1 to 100) of shares, and the server will remove that price from the price list. When the client sells, it sells all the shares of the stock, and the server will remove the price from the price list. Overall yield is calculated by . Z is the buying rate. X and Y are the selling rates. If the stock’s price increases by X or if the price decreases by Y, then the client will attempt to sell all the shares of that stock. In the client program, Z, X, and Y are set to 80, 1, and -2, respectively. The buying rate is set to 80% because we wanted the client to buy most of the times. Since we are using realistic historical stock prices, the prices only show small gradual increases and decreases (about ±0.05% between each adjacent prices). We decided to set X to 1% and Y to -2% so the client will attempt to sell within the 2100 *maxAttemptedTransactions*. The buy or sell transaction still has a 10% transaction fail rate that may occur at the server. Thus, the overall yield from the chosen X ,Y, and Z and price list is within 0-10% in the negative range.

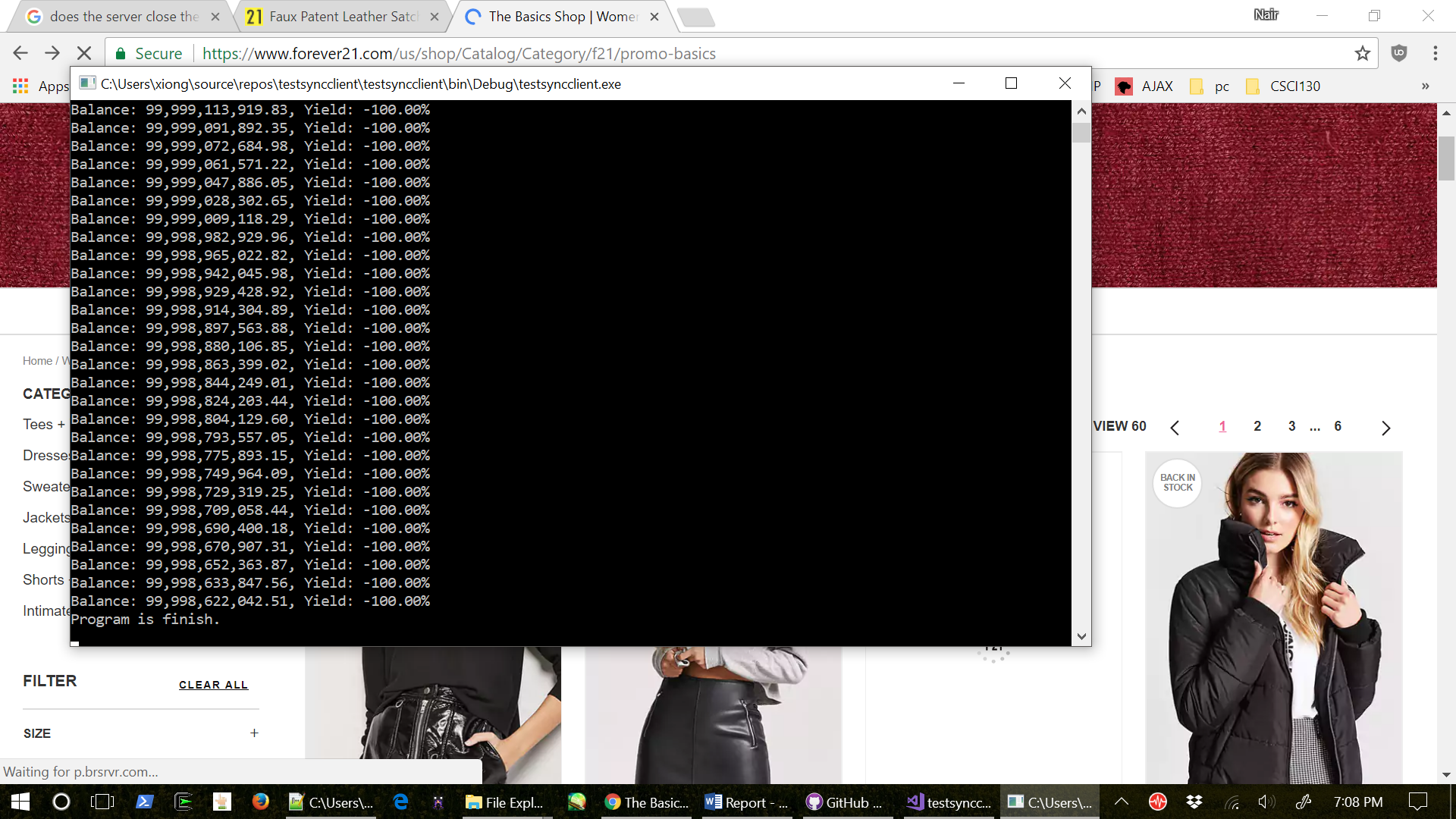
 

These two screenshots show the last few results of two different runs of *syncclient.exe*. From the two outputs, the yield is close to 0%, but in the negative range.

When X was set to 5% and Y was still -2%, the overall yield is more than -10%:



When X was set to 25% and Y was changed to -25%, the overall yield is -100%:



Most likely, sales never occurred because the stocks never increased or decreased by 25% within adjacent prices on the price list.

The automated stock trading program is written in C# and it includes the *server.cs*, *syncclient.cs*, and *Data.txt* files. Since the client only need to connect with the server, we simply use synchronous methods, whereas the server needed to talk to one or more clients, so asynchronous methods were use in the server program. After the server initializes the stock prices, the server begins to listen for connection requests. Once a client is connected to the server, the client and the server will continue to use the sockets on their program until the client have tried to make 2100 transactions. After 2100 attempted transactions are made, the client will close the socket. Thus, the server will also close the socket. Z is the buying rate, and X and Y together is the selling rate. If Z is high rate, then the client will buy frequently. If X is too high and Y is too low, then the client will not sell any. If the client buys too much, but sell only a few, the yield may be low (0±25%), but if the client buys too much and never sells the yield will be -100%. If the client buys at the same rate as it sells, then the yield will be probably be 0±25% since the client will not gain much from selling all the stocks at the next price. With the realistic prices, X and Y needs to low and Z needs to be high or moderate.