**Test Plan Report**

This is the test case report that have more detail than the test plan and have reason for each test case and the result pass and fail.

**Test01 :** CSV Reading

**Test Detail:** reading csv by the c++ using the getline() method and delimitator is “,”

**Result : Pass**

**Test02 :** Socket Test (one way communication)

**Test Detail:** socket communication is establish between two c++ file. One as listener and one as sender. Sender require to send a string “hello world” to the listener

**Result : Pass**

**Test03 :** Socket Test (Two way communication)

**Test Detail:** Socket communication is establish between two c++ file. Both c++ file compile with listener function and sender function so they can listen to each other while also send a string “hello server” and “hello client” to each other.

**Result : Pass**

**Test04 :** Socket Test (multiple message)

**Test Detail:** After both prompt can communicate with each other, now need to communicate more such as one connection (without while loop or for loop) send multiple time of messages. This is fail due to the code is well structured and no bug, but each time socket only can send one entire 1024 bit of string

**Result : Fail**

**Test05 :** Socket Test Socket Test (multiple access)

**Test Detail:** One server been established, while client.exe and client2.exe connect to the server and ask for reply “hello client”

**Result : Pass**

**Test06 :** Socket Test (large amount receiving)

**Test Detail:** More than 1024bit been receive from the client and end up it fail

**Solution:** Client can send one or more message compare to send a long string

**Result : Fail**

**Test07 :** Socket Test (large amount sending)

**Test Detail:** More than 1024bit been receive from the client and end up it fail

**Solution:** Server can send one or more message compare to send a long string

**Result : Fail**

**Test08 :** Socket Test (client view the stock detail)

**Test Detail:** Client can press “1” and send to server while server send back the stock detail that read by the csv file

**Result : Pass**

**Test09 :** Socket Test (client want to sell order via socket)

**Test Detail:** Client can press “2” and continue to put the desire stock symbol that want to sell and quantity and price and server able to server such information before process the order

**Result : Pass**

**Test10 :** Socket Test (client want to buy order via socket)

**Test Detail:** Client can press “3” and continue to put the desire stock symbol that want to buy and quantity and price and server able to server such information before process the order

**Result : Pass**

**Test11:** Socket Test (client want to buy 5 orders via socket)

**Test Detail:** By running the buyspammer.exe, server able to receive 5 different order from the username=”tester” before process the order

**Result : Pass**

**Test12:** Socket Test (client want to sell 5 orders via socket)

**Test Detail:** By running the sellspammer.exe, server able to receive 5 different order from the username=”tester” before process the order

**Result : Pass**

**Test13: Algorithm Test** (initial matching order)

**Test Detail:** Initially the algorithm doesn’t need to take in order and using the unordermap (hash map) to process each of the order and do the matching if there is same symbol and price then it will be match. But due to the unordermap structure it always fail to match the symbol and price and there will be memory outflow or overflow. Hence, give up using the unordermap and this test case was fail

**Solution:** new matching algorithm been made and complete solve this problem

**Result : Fail**

**Test14: Algorithm Test** (updated matching order : matchBuyOrders)

**Test Detail:** The algorithm need to have a order type data as parameter and it will prioritize to take symbol and price match the pendingSellOrder (with acending looping), smaller the orderid will been process and matching to the parameter order first.

**Result : Pass**

**Test15: Algorithm Test** (updated matching order : matchSellOrders)

**Test Detail:** The algorithm need to have a order type data as parameter and it will prioritize to take symbol and price match the pendingBuyOrders (with acending looping), smaller the orderid will been process and matching to the parameter order first.

**Result : Pass**

**Test16: Algorithm Test** (View All Pending Stock, Action=4)

**Test Detail:** The algorithm need to loop through the pendingBuyOrders and pendingSellOrders and display out the orderid, username, stock symbol, price and quantity

**Result : Pass**

**Test17: Algorithm Test** (View Holding (client holding), Action=5)

**Test Detail:** The algorithm need to loop through the holding vector at the client.cpp and display it out

**Result : Pass**

**Test18: Quit Program**

**Test Detail:** The program able to exit when the user prompt in “quit”

**Result : Pass**