User Manual

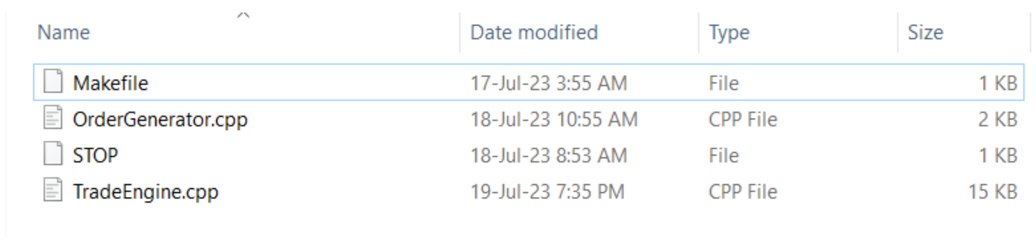
Trade Engine

Developed By
ISHARA THILINA

Table of Contents

Program configuration

All source files are included in the source directory. You can see them inside as follows:



Name	Date modified	Type	Size
Makefile	17-Jul-23 3:55 AM	File	1 KB
OrderGenerator.cpp	18-Jul-23 10:55 AM	CPP File	2 KB
STOP	18-Jul-23 8:53 AM	File	1 KB
TradeEngine.cpp	19-Jul-23 7:35 PM	CPP File	15 KB

Figure 1: Source Directory

TradeEngine.cpp is the main trade engine program. It runs as a TCP/IP server.

You can change the trade engine's running server port by modifying the value of *#define SERVER_PORT 5051* in the TradeEngine.cpp file. It's the port for receiving order data into TradeEngine.

```
#define SERVER_PORT 5051
```

TradeEngine has another port for sending matched trade data. You can view it as:

```
#define SERVER_PORT_SEND_DATA 5052
```

- If a client is connected to the *"SERVER_PORT_SEND_DATA"* port, TradeEngine will send matched trade data through that connection. If not, TradeEngine will save the matched trade data in the "**Completed_Trade_Data.log**" file.

OrderGenerator.cpp is a random order generator program used to test the high-volume order behavior in our main TradeEngine program. It runs as a client socket program. To configure it, you need to set the TradeEngine server IP and port by modifying the following values in the OrderGenerator.cpp file:

```
#define SERVER_IP "127.0.0.1"
#define SERVER_PORT 5051
```

Additionally, you can control the order generation interval using the following value:

```
#define MSG_INTERVEL 100
```

- If the *MSG_INTERVAL* value is set to 100, it means the OrderGenerator program generates random orders every 100 milliseconds. Otherwise, it generates 10 orders per second (10 TPS). You can increase the order generation TPS by decreasing the *MSG_INTERVAL* value.

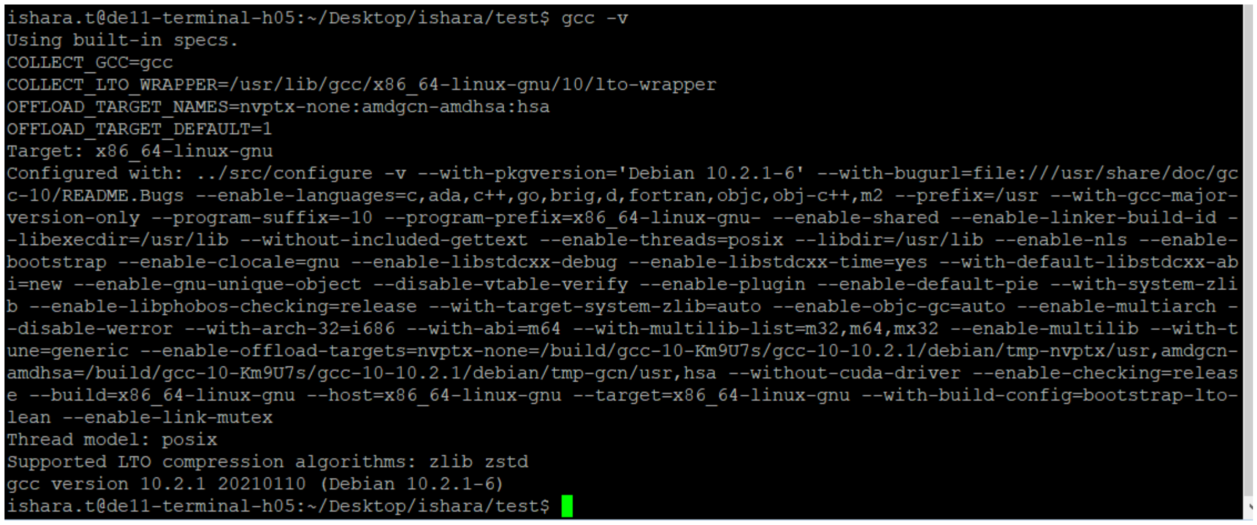
Program Compilation

To compile the program, you need to have the GCC compiler with build essentials installed on your PC. You can verify this by running the command "**gcc -v**". If you have GCC compiler with build essentials, it will give output similar to the following:

gcc version 10.2.1 20210110 (Debian 10.2.1-6)
Distributor ID: Debian
Description: Debian GNU/Linux 11 (bullseye)
Release: 11

***This instruction manual and development testing were performed using that OS and GCC version specified*

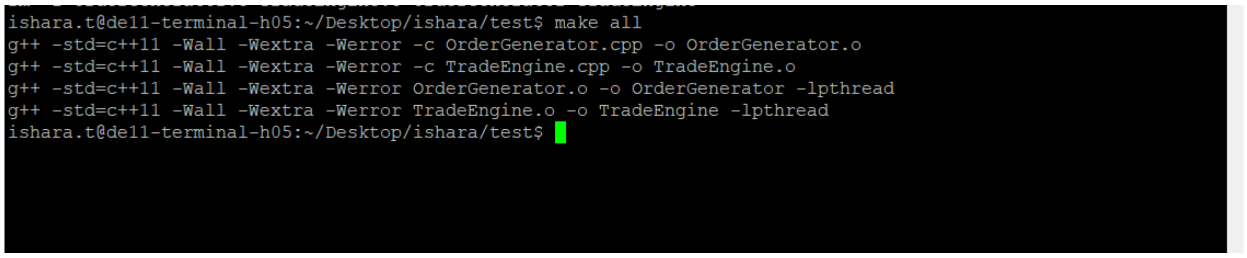
>_ gcc -v



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/10/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:amdgcn-amdhsa:hsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Debian 10.2.1-6' --with-bugurl=file:///usr/share/doc/gc
c-10/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,m2 --prefix=/usr --with-gcc-major-
version-only --program-suffix=-10 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id -
-libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-
bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-ab
i=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zli
b --enable-libphobos-checking=release --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch -
-disable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-t
une=generic --enable-offload-targets=nvptx-none=/build/gcc-10-Km9U7s/gcc-10-10.2.1/debian/tmp-nvptx/usr,amdgcn-
amdhsa=/build/gcc-10-Km9U7s/gcc-10-10.2.1/debian/tmp-gcn/usr,hsa --without-cuda-driver --enable-checking=releas
e --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu --with-build-config=bootstrap-lto-
lean --enable-link-mutex
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 10.2.1 20210110 (Debian 10.2.1-6)
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$
```

The program can be compiled by navigating to the **source** directory and running "**make all**" in your terminal.

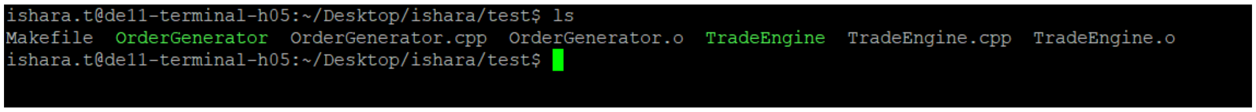
>_ make all



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ make all
g++ -std=c++11 -Wall -Wextra -Werror -c OrderGenerator.cpp -o OrderGenerator.o
g++ -std=c++11 -Wall -Wextra -Werror -c TradeEngine.cpp -o TradeEngine.o
g++ -std=c++11 -Wall -Wextra -Werror OrderGenerator.o -o OrderGenerator -lpthread
g++ -std=c++11 -Wall -Wextra -Werror TradeEngine.o -o TradeEngine -lpthread
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$
```

After successful compilation, you can see the executable files by running the "**ls**" command.

>_ ls

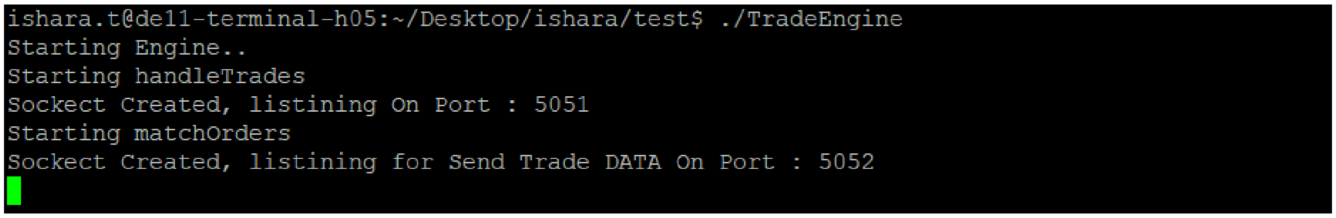


```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ ls
Makefile  OrderGenerator  OrderGenerator.cpp  OrderGenerator.o  TradeEngine  TradeEngine.cpp  TradeEngine.o
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$
```

Running TradeEngine and Program Behavior

To start the trade engine, run "**./TradeEngine**" in the terminal. It will begin displaying the following console output:

>_ ./TradeEngine



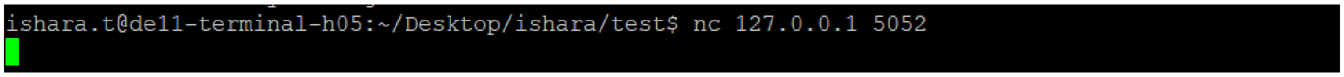
```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ ./TradeEngine
Starting Engine..
Starting handleTrades
Sockect Created, listining On Port : 5051
Starting matchOrders
Sockect Created, listining for Send Trade DATA On Port : 5052
```

Additionally, the TradeEngine creates a "**Completed_trade_Data.log**" log file in the same directory. This log file will include all completed trade details while the client is not connected to the send data port for receiving completed Trades. The Trade Engine console prints normal engine behaviors such as incoming orders and matching details in real time. However, for performance reasons, the program updates the "**Completed_trade_Data.log**" file every 3 seconds only. This is because handling high-speed trading requires certain resources, and performing frequent log I/O operations can impact performance. Instead, the Trade Engine program saves all trade details temporarily in a queue and prints that data every 3 seconds using another thread.(If the client is connected to the SERVER_PORT_SEND_DATA port, trade details are directly sent to the client, and that data is not included in the log.)

- After starting the TradeEngine, you can connect to SERVER_PORT to send order data, and you can connect to SERVER_PORT_SEND_DATA to receive matched trade data. If a client is unable to connect to SERVER_PORT_SEND_DATA, the **Completed_trade_Data.log** file will include the order data.

You can monitor SERVER_PORT_SEND_DATA using the terminal using the telnet or netcat command as well. After TradeEngine is started, you can do it as follows:

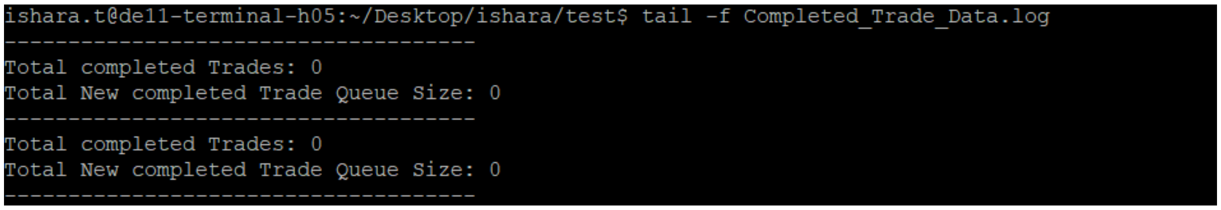
>_ nc 127.0.0.1 5052



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ nc 127.0.0.1 5052
```

you can use the "tail" command to monitor the "Completed_trade_Data.log" file and observe all offline Trade Engine behaviors. Run "**tail -f Completed_trade_Data.log**" in the terminal to achieve this.

>_ tail -f Completed_trade_Data.log



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ tail -f Completed_Trade_Data.log
-------------------------------------
Total completed Trades: 0
Total New completed Trade Queue Size: 0
-------------------------------------
Total completed Trades: 0
Total New completed Trade Queue Size: 0
-------------------------------------
```

Total completed Trades: *That describes the number of trades completed when offline after the engine start.*

Total New completed Trade Queue Size: *That describes the number of offline trades completed during idle time in the log (3 seconds). If there are trades in the queue, you can view their details here*.

Order Format

You can manually send orders to the trade engine using the **telnet** or **nc** command, or any other relevant socket data writing application. Alternatively, you can use the automatic random order generation provided by our OrderGenerator program.

If you want to send a manual order into the engine, the data string must be in the following format:

ID,Type,Price,Quantity
**The values should be separated by commas.*

ID: A unique identifier for the order.
Type: Buy or sell.
Price: The price at which the buyer is willing to buy or the seller is willing to sell.
Quantity: The quantity of shares the buyer wants to buy or the seller wants to sell.

For example: **8801,sell,94.000000,2**

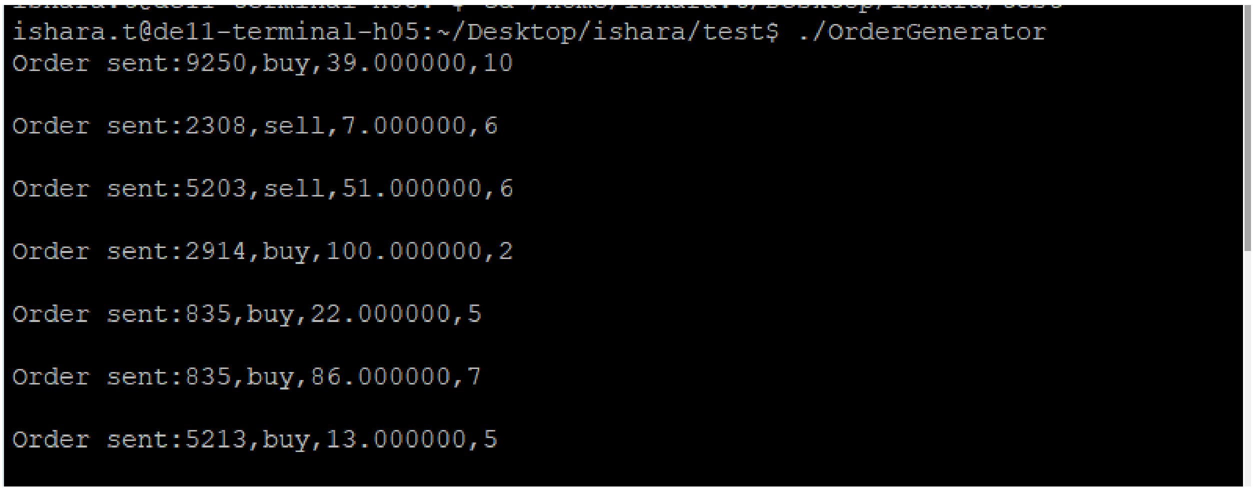
More details on manual order sending will be demonstrated in the "Testing trade engine" section.

Running Programs

If the trade engine is running, you can execute the random order generator to observe the entire process of order generation, sending, trade engine matching, and matched trade data sending or completed trade logging.

To run the order generator program, enter "**./OrderGenerator**" in your terminal.
You will then see the order generator running as shown below:

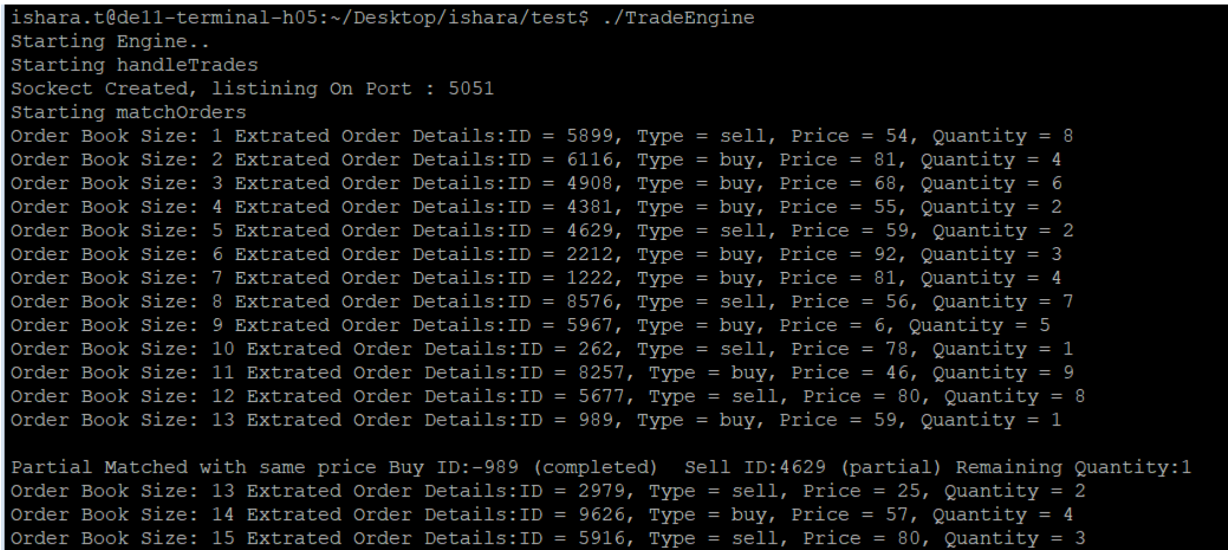
>_ ./OrderGenerator



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ ./OrderGenerator
Order sent:9250,buy,39.000000,10

Order sent:2308,sell,7.000000,6

Order sent:5203,sell,51.000000,6

Order sent:2914,buy,100.000000,2

Order sent:835,buy,22.000000,5

Order sent:835,buy,86.000000,7

Order sent:5213,buy,13.000000,5
```

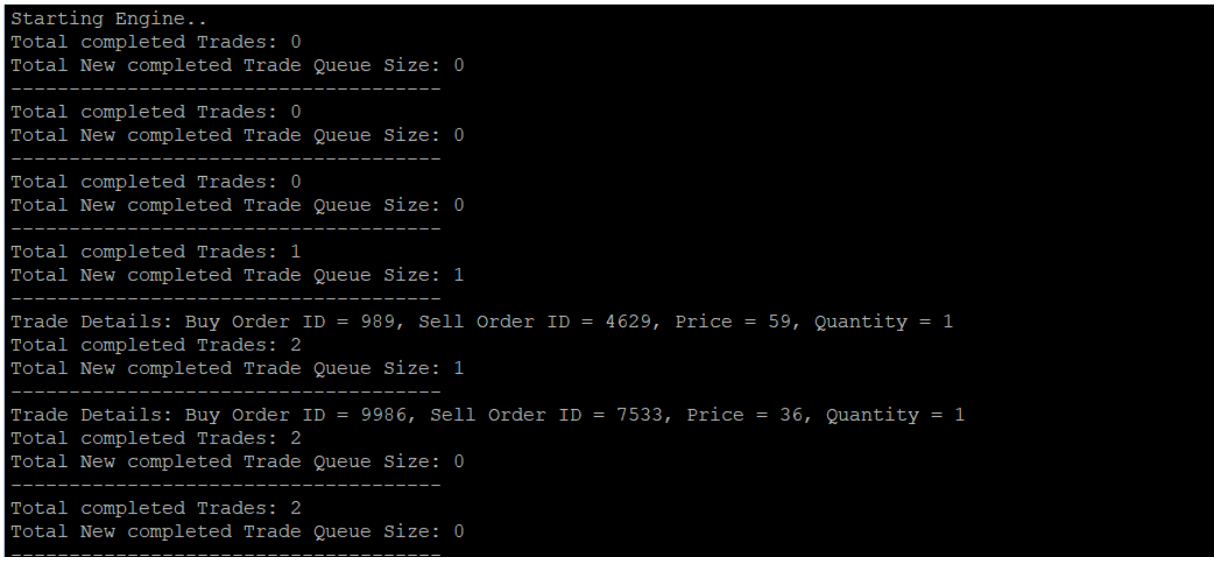
If You are in offline mode, Then the trade engine and the "**Completed_trade_Data.log**" exhibit the following behavior:

>_ ./TradeEngine



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ ./TradeEngine
Starting Engine..
Starting handleTrades
Sockect Created, listining On Port : 5051
Starting matchOrders
Order Book Size: 1 Extrated Order Details:ID = 5899, Type = sell, Price = 54, Quantity = 8
Order Book Size: 2 Extrated Order Details:ID = 6116, Type = buy, Price = 81, Quantity = 4
Order Book Size: 3 Extrated Order Details:ID = 4908, Type = buy, Price = 68, Quantity = 6
Order Book Size: 4 Extrated Order Details:ID = 4381, Type = buy, Price = 55, Quantity = 2
Order Book Size: 5 Extrated Order Details:ID = 4629, Type = sell, Price = 59, Quantity = 2
Order Book Size: 6 Extrated Order Details:ID = 2212, Type = buy, Price = 92, Quantity = 3
Order Book Size: 7 Extrated Order Details:ID = 1222, Type = buy, Price = 81, Quantity = 4
Order Book Size: 8 Extrated Order Details:ID = 8576, Type = sell, Price = 56, Quantity = 7
Order Book Size: 9 Extrated Order Details:ID = 5967, Type = buy, Price = 6, Quantity = 5
Order Book Size: 10 Extrated Order Details:ID = 262, Type = sell, Price = 78, Quantity = 1
Order Book Size: 11 Extrated Order Details:ID = 8257, Type = buy, Price = 46, Quantity = 9
Order Book Size: 12 Extrated Order Details:ID = 5677, Type = sell, Price = 80, Quantity = 8
Order Book Size: 13 Extrated Order Details:ID = 989, Type = buy, Price = 59, Quantity = 1

Partial Matched with same price Buy ID:-989 (completed)  Sell ID:4629 (partial) Remaining Quantity:1
Order Book Size: 13 Extrated Order Details:ID = 2979, Type = sell, Price = 25, Quantity = 2
Order Book Size: 14 Extrated Order Details:ID = 9626, Type = buy, Price = 57, Quantity = 4
Order Book Size: 15 Extrated Order Details:ID = 5916, Type = sell, Price = 80, Quantity = 3
```

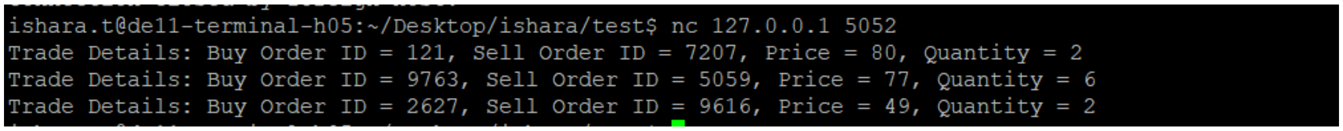
>_ tail -f Completed_trade_Data.log



```
Starting Engine..
Total completed Trades: 0
Total New completed Trade Queue Size: 0
-------------------------------------
Total completed Trades: 0
Total New completed Trade Queue Size: 0
-------------------------------------
Total completed Trades: 0
Total New completed Trade Queue Size: 0
-------------------------------------
Total completed Trades: 1
Total New completed Trade Queue Size: 1
-------------------------------------
Trade Details: Buy Order ID = 989, Sell Order ID = 4629, Price = 59, Quantity = 1
Total completed Trades: 2
Total New completed Trade Queue Size: 1
-------------------------------------
Trade Details: Buy Order ID = 9986, Sell Order ID = 7533, Price = 36, Quantity = 1
Total completed Trades: 2
Total New completed Trade Queue Size: 0
-------------------------------------
Total completed Trades: 2
Total New completed Trade Queue Size: 0
-------------------------------------
```

If you connect to the SERVER_PORT_SEND_DATA port, you can view matched order data as shown below

>_ nc 127.0.0.1 5052



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ nc 127.0.0.1 5052
Trade Details: Buy Order ID = 121, Sell Order ID = 7207, Price = 80, Quantity = 2
Trade Details: Buy Order ID = 9763, Sell Order ID = 5059, Price = 77, Quantity = 6
Trade Details: Buy Order ID = 2627, Sell Order ID = 9616, Price = 49, Quantity = 2
```

- To stop the TradeEngine and OrderGenerator, run the "**STOP**" script by executing "**./STOP**" in the terminal. (*You may need to set execute permissions to run the STOP script.*
  *You can use the **"chmod 777 STOP"** command to grant all permissions*)

Testing Trade Engine

To test different scenarios in our TradeEngine, we need to run the TradeEngine and send manual orders. Telnet commands can be used for sending manual orders to the TradeEngine.

Price and Quantity Same Orders Matching Scenarios

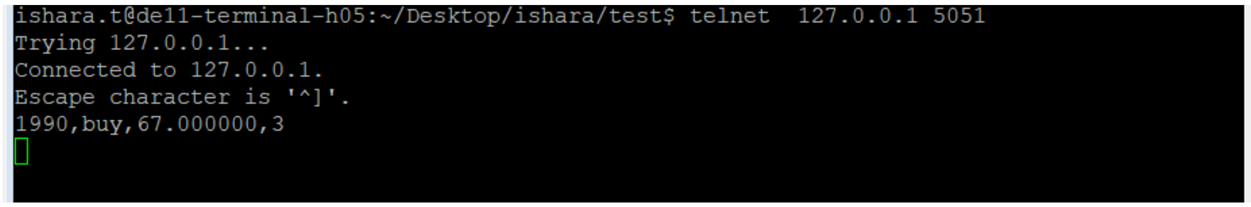
Follow these steps to test price and quantity same buy and sell order matching scenarios:

1. Run the TradeEngine and tail the "Completed_Trade_Data.log" file.
2. Use Telnet to connect to the TradeEngine and send the following test order data:

   1990,buy,67.000000,3

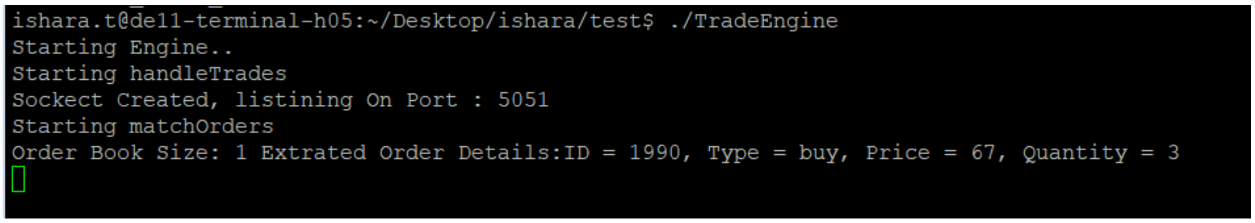
Then observe the behavior in the TradeEngine, Completed_Trade_Data.log, and Telnet terminal.

>_ telnet 127.0.0.1 5051



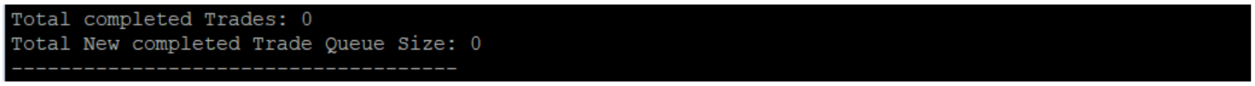
```
ishara.t@de11-terminal-h05:~/Desktop/ishara/test$ telnet  127.0.0.1 5051
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
1990,buy,67.000000,3
```

>_ ./TradeEngine



```
ishara.t@de11-terminal-h05:~/Desktop/ishara/test$ ./TradeEngine
Starting Engine..
Starting handleTrades
Sockect Created, listining On Port : 5051
Starting matchOrders
Order Book Size: 1 Extrated Order Details:ID = 1990, Type = buy, Price = 67, Quantity = 3
```

>_ tail -f Completed_trade_Data.log



```
Total completed Trades: 0
Total New completed Trade Queue Size: 0
-------------------------------------
```

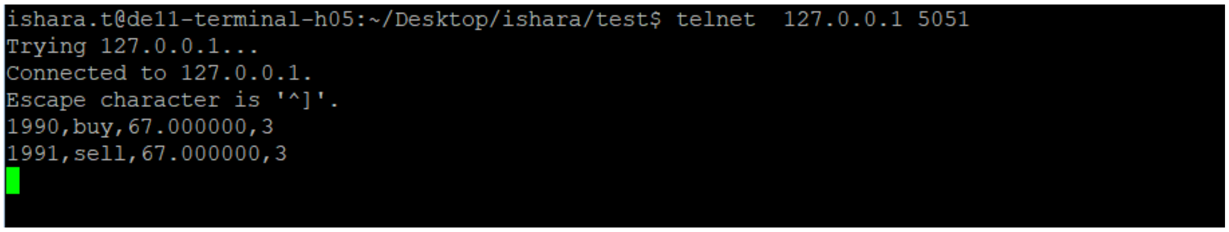
- ➢ You can see the orders received by the TradeEngine and the corresponding extracted details. However, in the Completed_Trade_Data.log, you won't see anything because the order has not been matched yet.

3. Send the following order into the TradeEngine:

   1991,sell,67.000000,3

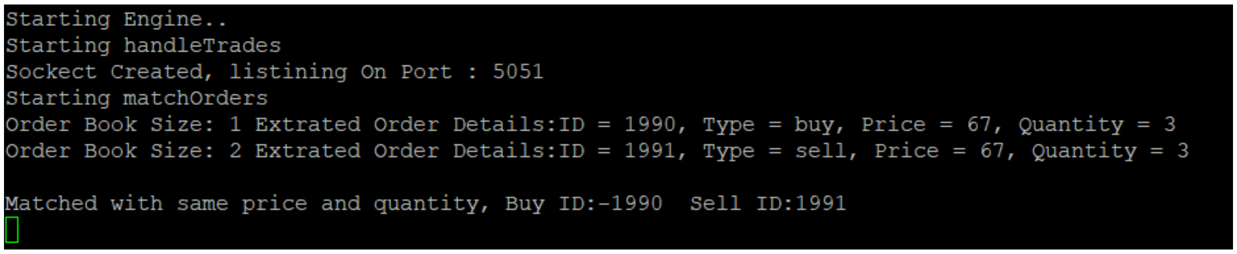
Now observe the behavior in the TradeEngine, Completed_Trade_Data.log, and Telnet terminal.

>_ telnet 127.0.0.1 5051



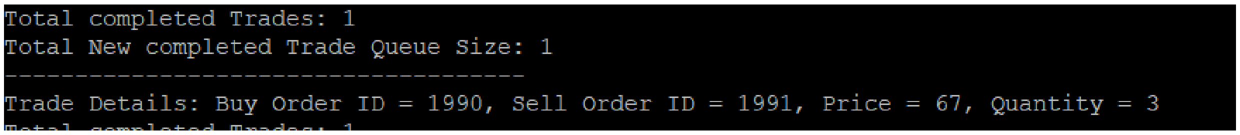
```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ telnet  127.0.0.1 5051
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
1990,buy,67.000000,3
1991,sell,67.000000,3
```

>_ ./TradeEngine



```
Starting Engine..
Starting handleTrades
Sockect Created, listining On Port : 5051
Starting matchOrders
Order Book Size: 1 Extrated Order Details:ID = 1990, Type = buy, Price = 67, Quantity = 3
Order Book Size: 2 Extrated Order Details:ID = 1991, Type = sell, Price = 67, Quantity = 3

Matched with same price and quantity, Buy ID:-1990  Sell ID:1991
```

>_ tail -f Completed_trade_Data.log

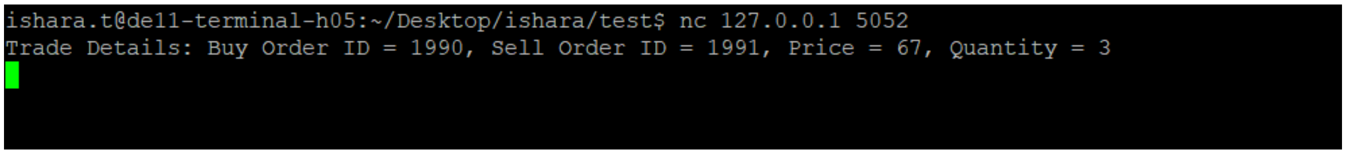


```
Total completed Trades: 1
Total New completed Trade Queue Size: 1
-------------------------------------
Trade Details: Buy Order ID = 1990, Sell Order ID = 1991, Price = 67, Quantity = 3
```

- You can see the received order being matched with a previous order. The TradeEngine console will print: "**Matched with same price and quantity, Buy ID: 1990, Sell ID: 1991**". Additionally, in the Completed_Trade_Data.log, you will find the matched trade details printed as: "**Trade Details: Buy Order ID = 1990, Sell Order ID = 1991, Price = 67, Quantity = 3**". And The "**Total completed Trades**" will be updated to 1.

If you are connected to SERVER_PORT_SEND_DATA as well, you can observe the terminal output as shown below, and then the log file will be empty.

>_ nc 127.0.0.1 5052



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ nc 127.0.0.1 5052
Trade Details: Buy Order ID = 1990, Sell Order ID = 1991, Price = 67, Quantity = 3
```

- For ease of documenting and understanding without hassle, this user manual will show output as screenshots of **Completed_trade_Data.log** only. However, you can use either of the following commands to view TradeEngine output:

  >_ nc 127.0.0.1 5052
  >_ tail -f Completed_trade_Data.log

Additional Example

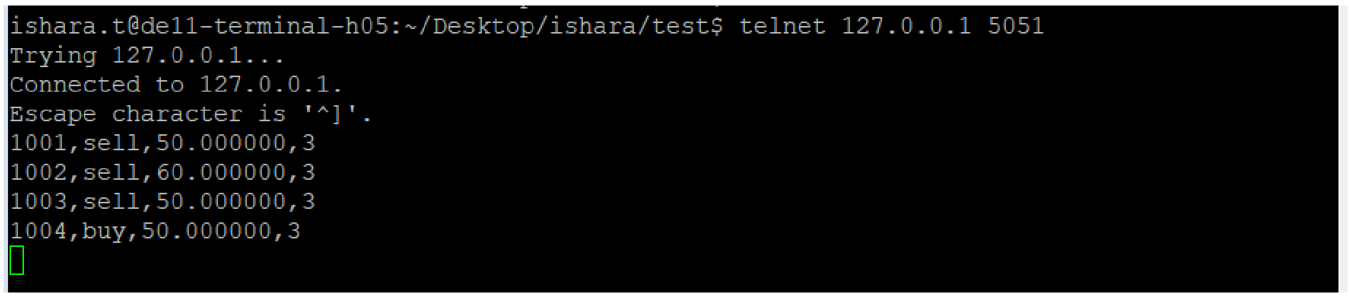
Continue following the above steps and test various order combinations and their behaviors.
another example here:

Start the TradeEngine and send the following orders to the TradeEngine.

1001,sell,50.000000,3
1002,sell,60.000000,3
1003,sell,50.000000,3
1004,buy,50.000000,3

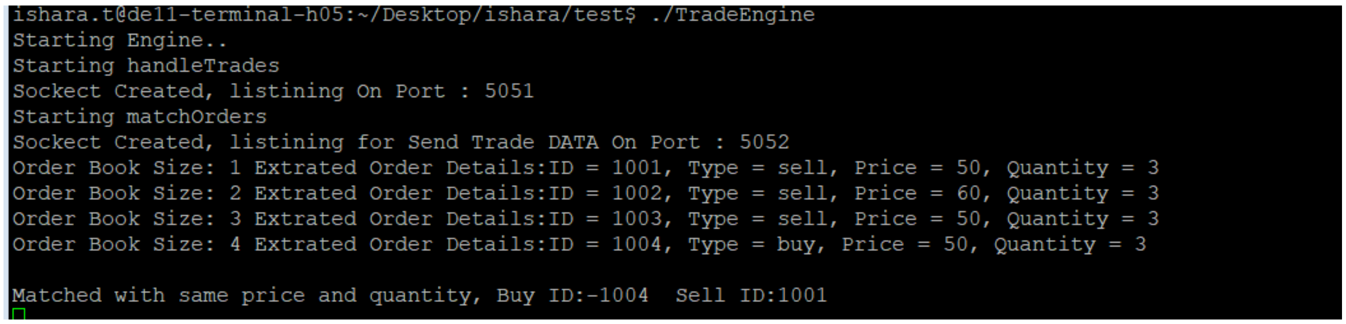
Then you can observe the following behavior in TradeEngine, Completed_Trade_Data.log, and the telnet terminal

>_ telnet 127.0.0.1 5051



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ telnet 127.0.0.1 5051
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
1001,sell,50.000000,3
1002,sell,60.000000,3
1003,sell,50.000000,3
1004,buy,50.000000,3
```

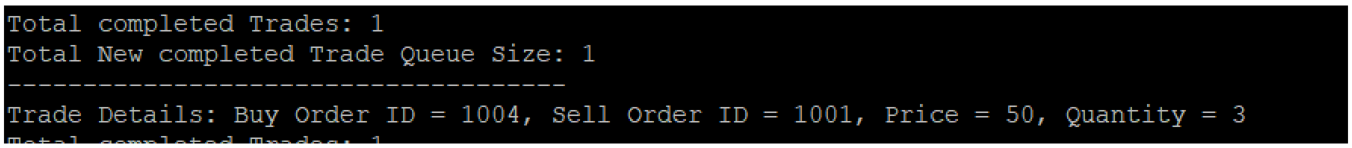
>_ ./TradeEngine



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ ./TradeEngine
Starting Engine..
Starting handleTrades
Sockect Created, listining On Port : 5051
Starting matchOrders
Sockect Created, listining for Send Trade DATA On Port : 5052
Order Book Size: 1 Extrated Order Details:ID = 1001, Type = sell, Price = 50, Quantity = 3
Order Book Size: 2 Extrated Order Details:ID = 1002, Type = sell, Price = 60, Quantity = 3
Order Book Size: 3 Extrated Order Details:ID = 1003, Type = sell, Price = 50, Quantity = 3
Order Book Size: 4 Extrated Order Details:ID = 1004, Type = buy, Price = 50, Quantity = 3

Matched with same price and quantity, Buy ID:-1004  Sell ID:1001
```

- You can see that the last Buy order ID 1004 matched with Order ID 1001, and Order ID 1003 also has the same price and quantity. However, TradeEngine matched it with 1001 because the **required system should prioritize early orders**.

>_ tail -f Completed_trade_Data.log



```
Total completed Trades: 1
Total New completed Trade Queue Size: 1
-------------------------------------
Trade Details: Buy Order ID = 1004, Sell Order ID = 1001, Price = 50, Quantity = 3
```

Partial Matching (Price Same and Different Quantity) Order Matching Scenarios

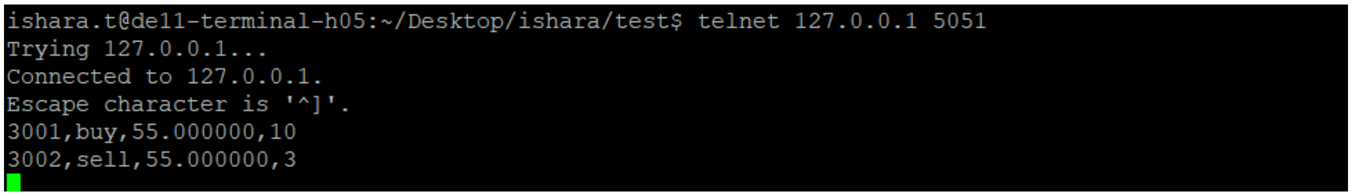
You can follow the previous steps and send the following orders to demonstrate partial match scenarios:

Start the engine and send the following orders into the TradeEngine:

3001,buy,55.000000,10
3002,sell,55.000000,3
3003,sell,55.000000,5
3004,sell,55.000000,2

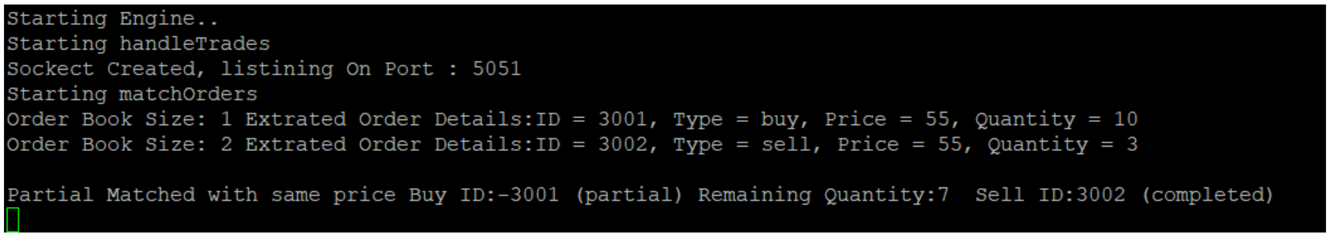
- When you add the data of the first two orders, you can see a partial match, and **order 3001 will have 7 quantities remaining**

>_ telnet 127.0.0.1 5051



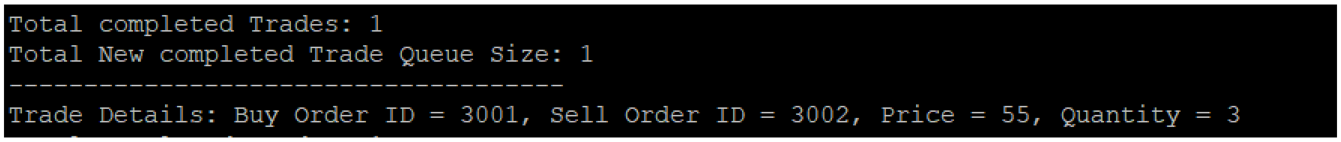
```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ telnet 127.0.0.1 5051
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
3001,buy,55.000000,10
3002,sell,55.000000,3
```

>_ ./TradeEngine



```
Starting Engine..
Starting handleTrades
Sockect Created, listining On Port : 5051
Starting matchOrders
Order Book Size: 1 Extrated Order Details:ID = 3001, Type = buy, Price = 55, Quantity = 10
Order Book Size: 2 Extrated Order Details:ID = 3002, Type = sell, Price = 55, Quantity = 3

Partial Matched with same price Buy ID:-3001 (partial) Remaining Quantity:7  Sell ID:3002 (completed)
```

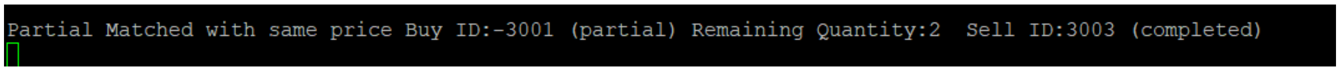
>_ tail -f Completed_trade_Data.log



```
Total completed Trades: 1
Total New completed Trade Queue Size: 1
------------------------------------
Trade Details: Buy Order ID = 3001, Sell Order ID = 3002, Price = 55, Quantity = 3
```

Now you can send order **3003,sell,55.000000,5**
After sending the order, TradeEngine will produce the following output:

>_ ./TradeEngine

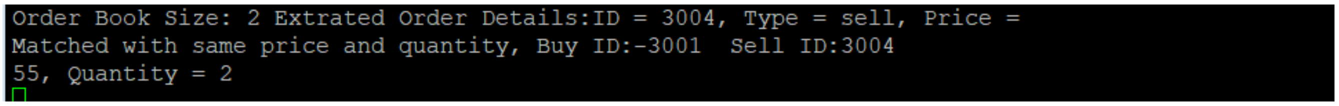


```
Partial Matched with same price Buy ID:-3001 (partial) Remaining Quantity:2  Sell ID:3003 (completed)
```

- You can see Still order 3001 having 2 remaining quantity

Now If you send the following order: **3004,sell,55.000000,2**
you will see that it matches with order 3001.

>_ ./TradeEngine



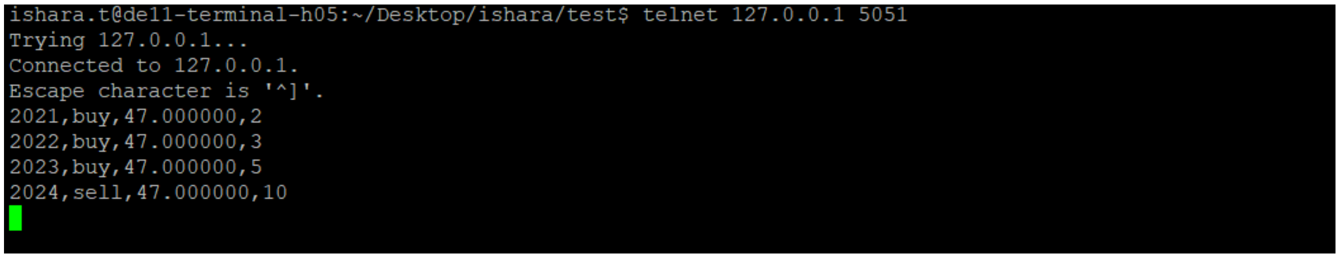
```
Order Book Size: 2 Extrated Order Details:ID = 3004, Type = sell, Price =
Matched with same price and quantity, Buy ID:-3001  Sell ID:3004
55, Quantity = 2
```

Additional Example

Let's consider another partial matching scenario like this
Please send the following orders to TradeEngine:

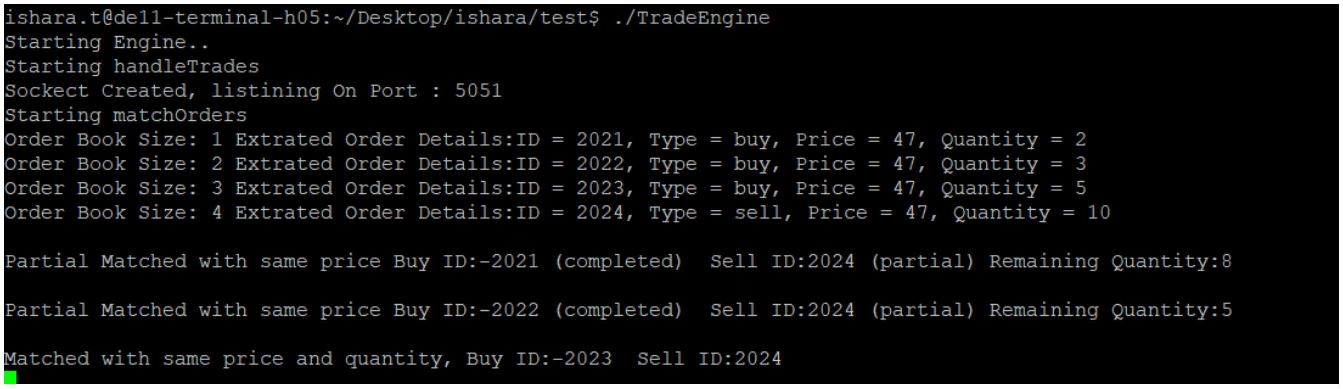
2021,buy,47.000000,2
2022,buy,47.000000,3
2023,buy,47.000000,5
2024,sell,47.000000,10

>_ telnet 127.0.0.1 5051



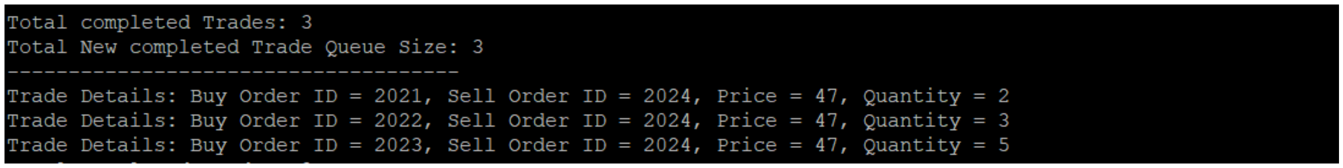
```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ telnet 127.0.0.1 5051
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
2021,buy,47.000000,2
2022,buy,47.000000,3
2023,buy,47.000000,5
2024,sell,47.000000,10
```

>_ ./TradeEngine



```
ishara.t@dell-terminal-h05:~/Desktop/ishara/test$ ./TradeEngine
Starting Engine..
Starting handleTrades
Sockect Created, listining On Port : 5051
Starting matchOrders
Order Book Size: 1 Extrated Order Details:ID = 2021, Type = buy, Price = 47, Quantity = 2
Order Book Size: 2 Extrated Order Details:ID = 2022, Type = buy, Price = 47, Quantity = 3
Order Book Size: 3 Extrated Order Details:ID = 2023, Type = buy, Price = 47, Quantity = 5
Order Book Size: 4 Extrated Order Details:ID = 2024, Type = sell, Price = 47, Quantity = 10

Partial Matched with same price Buy ID:-2021 (completed)  Sell ID:2024 (partial) Remaining Quantity:8

Partial Matched with same price Buy ID:-2022 (completed)  Sell ID:2024 (partial) Remaining Quantity:5

Matched with same price and quantity, Buy ID:-2023  Sell ID:2024
```

>_ tail -f Completed_trade_Data.log



```
Total completed Trades: 3
Total New completed Trade Queue Size: 3
-------------------------------------
Trade Details: Buy Order ID = 2021, Sell Order ID = 2024, Price = 47, Quantity = 2
Trade Details: Buy Order ID = 2022, Sell Order ID = 2024, Price = 47, Quantity = 3
Trade Details: Buy Order ID = 2023, Sell Order ID = 2024, Price = 47, Quantity = 5
```

- You can see that all Orders 20211, 2022, and 2023 are partially matched with Order 2024