Approach 01 - Using 2 apps (more real-world)

Trader App

* Reads the buy-sell records from 'order.csv' file
* Submit the orders to Exchange Application
* Receive the execution reports.
* Write the reports to 'execution_rep.csv'.

Exchange App

* Process buy-sell records
* Match orders
* Send execution reports to Trader app

* Real-world scenario
* Socket programming based

Approach 02 - Using 1 app

Exchange App

* Read buy-sell orders from 'order.csv'
* Process them and creak an execution report
* Write to the CSV file

- In this assignment, the main focus is how we create the matching logic.

Order.csv

Field Name	Type	Possible Values	Mandatory	Notes
Client Order ID	String	Alpha numeric string (max 7 chars)	Yes	This unique ID identifies the submitted order
Instrument	String	{Rose, Lavender, Lotus, Tulip, Orchid)	Yes	We will limit the instruments for these 5 types only
Side	Int	1 : Buy ; 2 : Sell	Yes	Specifies if the input order is a buy order or a sell order
Price	double	Price > 0.0	Yes	Price of one unit
Quantity	int	{10, 20, 30, …. , 1000} Order size must be a a multiple of 10. Min 10, Max 1000	Yes	Quantity of the order

Execution report .csv

Field Name	Type	Possible Values	Mandatory	Notes
Client Order ID	String	Alpha numeric string (max 7 chars)	Yes	This is the Client Order ID of the submitted order
Order ID	String	Alpha numeric string	Yes	System generated unique order ID
Instrument	String	{Rose, Lavender, Lotus, Tulip, Orchid)	Yes	We will limit the instruments for these 5 types only
Side	Int	1 : Buy ; 2 : Sell	Yes	Specifies if the order is a buy order or a sell order
Price	double	Price > 0.0	Yes	Price of one unit
Quantity	int	{10, 20, 30, …. , 1000} Order size must be a a multiple of 10. Min 10, Max 1000	Yes	Quantity of the order
Status	int	0 – New 1 – Rejected 2 – Fill 3 - Pfill	Yes	The status of the execution report
Reason	String	Max 50 chars	No	Contains the reject reason, when an order is not accepted into the system due to validation failure
Transaction Time	**String**	**YYYYMMDD-HHMMSS.sss**	**Yes**	**Every execution report should have the transaction time in the given format. This data can be used to check the speed and optimize your code**

* To handle the logic we need an order book. The orders get placed in the order book, and that produces the execution report.

1 book per instrument

Orderbook : Rose (Initially empty)

Order ID	Qty	Price	Price	Qty	Order ID

Buy

Sell

Orders.csv

ClientOrderID	Instrument	Side	Quantity	Price
aa13	Rose	2	100	65
aa14	Lavender	1	100	55

Rose

Order ID	Qty	Price	Price	Qty	Order ID
			65	100	ord1

Lavendar

Order ID	Qty	Price	Price	Qty	Order ID
ord2	100	55			

Execution Report.csv

Order ID	Cl. Order ID	Instrument	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	2	NEW	100	65
ord2	aa14	Lavendar	1	NEW	100	55

* Sellers with lower prices are more attractive.
  ∴ Higher priority.

* Buyers with higher prices are more attractive.
  ∴ Higher priority

- Fill orders (agressive order)
  * When there's a matching order (price, quantity) those orders don't go to the order book. They get executed directly. (exec_status = Fill)

- The order, these 2 records get placed in the exec_report doesn't matter

- Pfill orders

Orders.csv

ClientOrderID	Instrument	Side	Quantity	Price
aa13	Rose	2	100	55
aa14	Rose	2	100	45
aa15	Rose	1	200	45
aa16	Rose	2	100	45

Rose

Order ID	Qty	Price	Price	Qty	Order ID
			45	100	ord2
			55	100	ord1

This gets matched partially
∴

Execution Report.csv

Order ID	Cl. Order ID	Instrument	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	2	NEW	100	55
ord2	aa14	Rose	2	NEW	100	45
ord2	aa14	Rose	2	FILL	100	45
ord4	aa15	Rose	1	PFILL	100	45

Orders.csv

ClientOrderID	Instrument	Side	Quantity	Price
aa13	Rose	2	100	55
aa14	Rose	2	100	45
aa15	Rose	1	200	45
aa16	Rose	2	100	45

Rose

Order ID	Qty	Price	Price	Qty	Order ID
			55	100	ord1

Execution Report.csv

Order ID	Cl. Order ID	Instrument	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	2	NEW	100	55
ord2	aa14	Rose	2	NEW	100	45
ord2	aa14	Rose	2	FILL	100	45
ord4	aa15	Rose	1	PFILL	100	45
ord4	aa15	Rose	1	FILL	100	45
ord5	aa16	Rose	2	FILL	100	45

These 2 got matched.

ClientOrderID	Instrument	Side	Quantity	Price
aa13	Rose	2	100	55
aa14	Rose	2	100	45
aa15	Rose	1	200	45
aa16	Rose	2	100	45

Rose

Order ID	Qty	Price	Price	Qty	Order ID
aa14	100	45	55	100	ord1

有为 These 2 won't match. Seller has a high price

Execution Report.csv

Order ID	Cl. Order ID	Instrument	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	2	NEW	100	55
ord2	aa14	Rose	2	NEW	100	45
ord2	aa14	Rose	2	FILL	100	45
ord4	aa15	Rose	1	PFILL	100	45

** when there's no records to match price exactly, the seller order will go to the most attractive buyer. (one in the top)

** The execution happens at the price which is already in the order book.

aa14	Rose	1	100	65
aa15	Rose	2	300	1
aa16	Rose	1	100	200

Rose

Order ID	Qty	Price	Price	Qty	Order ID
			1	100	aa15

Execution Report.csv

Order ID	Cl. Order ID	Instrument	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	1	NEW	100	55
ord2	aa14	Rose	1	NEW	100	65
ord2	aa14	Rose	1	FILL	100	65
ord3	aa15	Rose	2	PFILL	100	65
ord1	aa13	Rose	1	FILL	100	55
ord3	aa15	Rose	2	PFILL	100	55
ord3	aa15	Rose	1	FILL	100	1

Example

Orders.csv

ClientOrderID	Instrument	Side	Quantity	Price
aa13	Rose	1	100	55
aa14	Rose	1	100	65
aa15	Rose	2	300	1
aa16	Rose	1	100	2

Rose

Order ID	Qty	Price	Price	Qty	Order ID
aa13	100	65			
aa13	100	55			

Execution Report.csv

Order ID	Cl. Order ID	Instrumen	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	1	NEW	100	55
ord2	aa14	Rose	1	NEW	100	65
ord2	aa14	Rose	1	FILL	100	65
ord3	aa15	Rose	2	PFILL	100	65
ord1	aa13	Rose	1	FILL	100	55
ord3	aa15	Rose	2	PFILL	100	55
ord3	aa15	Rose	2	FILL	100	1
ord4	aa16	Rose	1	FILL	100	1

ClientOrderID	Instrument	Side	Quantity	Price
aa13	Rose	1	100	55
aa14	Rose	1	100	65
aa15	Rose	2	300	1
aa16	Rose	1	100	2

Rose

Order ID	Qty	Price	Price	Qty	Order ID
aa13	100	55			

Execution Report.csv

Order ID	Cl. Order ID	Instrumen	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	1	NEW	100	55
ord2	aa14	Rose	1	NEW	100	65
ord2	aa14	Rose	1	FILL	100	65
ord3	aa15	Rose	2	PFILL	100	65
ord1	aa13	Rose	1	FILL	100	55
ord3	aa15	Rose	2	PFILL	100	55
ord3	aa15	Rose	2	FILL	100	1
ord4	aa16	Rose	1	FILL	100	1

Orders.csv

ClientOrderID	Instrument	Side	Quantity	Price
aa13	Rose	1	100	55
aa14	Rose	1	100	65
aa15	Rose	2	300	1
aa16	Rose	1	100	2

Rose

Order ID	Qty	Price	Price	Qty	Order ID
			1	100	aa15

Execution Report.csv

Order ID	Cl. Order ID	Instrumen	Side	Exec Status	Quantity	Price
ord1	aa13	Rose	1	NEW	100	55
ord2	aa14	Rose	1	NEW	100	65
ord2	aa14	Rose	1	FILL	100	65
ord3	aa15	Rose	2	PFILL	100	65
ord1	aa13	Rose	1	FILL	100	55
ord3	aa15	Rose	2	PFILL	100	55
ord3	aa15	Rose	2	FILL	100	1
ord4	aa16	Rose	1	FILL	100	1

* If somebody wants to buy at a higher price than what is already in the order books, it's a matching

* When a buy order is there, if a seller arrives with ≥ price then it's also a match

```
struct Order {
    String ID;
    double Price;
    int quantity;
    State [0,1,2,3] state; => New by default
    String reason ("")

  Methods to compare for the set
}
```

```
class OrderBook
    set <Order> buy_orders
    set <Order> sell_orders

    Void addOrder(const Order &order)
    void DisplayOrders() {
```

```
class MatchingEngine
    MatchOrder(string & Line)
    private: validate order()
    PrintReport().

    5order book
    set <Orders: finalized_orders>
```

order.csv

↓ read line by line

Matching Engine

- Validate an order and return an Order
- Match the order

↙ else match with order.

↘ If invalid directly add to final

Monitor Signalling disciplines

01> Signal and exit : A process executing (inside a monitor) a signal on a condition queue is required to leave the monitor immediately after generating the signal.

02> Signal and wait : The process is not required to leave the monitor after executing the signal.
So the signalling process might change the state
Therefore,

use	`while (condition)wait();`
rather than	`if (condition) wait();`

03> Signal and continue.