+Read Me:

- Your task is to implement an in-memory cache of order objects that supports

adding new orders, removing existing orders and matching buy and sell orders.

- On "order" is a request to buy or sell a financial security (eg. bond, stock,

commodity, etc.)

- Each order is uniquely identified by an order id

- Each security has a different security id

- Order matching occurs for orders with the same security id, different side (buy or sell),

and different company (company of person who requested the order)

- Provide an implementation for the OrderCacheInterface class in OrderCache.h

- An Order class is provided for you:

- This class holds basic order info

- Do not remove the provided member variables and methods in the Order class

- You may add additional members if you like

- For your implementation of OrderCacheInterface:

- Write a class that derives OrderCacheInterface

- Choose appropriate data structure(s) to hold Order objects and any additional data you'd like

- Implement the following methods (do not change their signatures)

- addOrder()

- cancelOrder()

- cancelOrdersForUser()

- cancelOrdersForSecIdWithMinimumQty()

- getMatchingSizeForSecurity()

- getAllOrders()

- Add any additional methods and variables you'd like to your class

- There are more comments in OrderCache.h to provide additional guidance

- Submit all files as email attachments or as .zip file. You do not need to submit

a main() or an executable. We will build an executable using your submitted

code for implementing the OrderCacheInterface class to run tests

- You do not need to submit any test cases in your code or any test results. Though

I highly recommend you run various tests yourself for verification

- Use up to C++17. Your code must compile. Code should be platform agnostic.

- Single-threaded support is sufficient. Adding thread safety is not necessary but

would be viewed as extra credit.

- Order matching rules for getMatchingSizeForSecurity()

- Your implementation of getMatchingSizeForSecurity() should give the total qty that can match for a security id

- Can only match orders with the same security id

- Can only match a Buy order with a Sell order

- Buy order can match against multiple Sell orders (and vice versa)

- eg a security id "ABCD" has

Buy order with qty 10000

Sell order with qty 2000

Sell order with qty 1000

- security id "ABCD" has a total match of 3000. The Buy order's qty is big enough

to match against both Sell orders and still has 7000 remaining

- Any order quantity already allocated to a match cannot be reused as a match

against a differnt order (eg the qty 3000 matched above for security id "ABCD" example)

- Some orders may not match entirely or at all

- Users in the same company cannot match against each other

- Order matching example and explanation

- Example set of orders added using addOrder()

OrdId1 SecId1 Buy 1000 User1 CompanyA

OrdId2 SecId2 Sell 3000 User2 CompanyB

OrdId3 SecId1 Sell 500 User3 CompanyA

OrdId4 SecId2 Buy 600 User4 CompanyC

OrdId5 SecId2 Buy 100 User5 CompanyB

OrdId6 SecId3 Buy 1000 User6 CompanyD

OrdId7 SecId2 Buy 2000 User7 CompanyE

OrdId8 SecId2 Sell 5000 User8 CompanyE

- Explanation

- SecId1

- SecId1 has 1 Buy order and 1 Sell order

- Both orders are for users in CompanyA so they are not allowed to match

- There are no matches for SecId1

- SecId2

- OrdId2 matches quantity 600 against OrdId4

- OrdId2 matches quantity 2000 against OrdId7

- OrdId2 has a total matched quantity of 2600

- OrdId8 matches quantity 100 against OrdId5 only

- OrdId8 has a remaining qty of 4900

- OrdId4 had its quantity fully allocated to match OrdId2

- No remaining qty on OrdId4 for the remaining 4900 of OrdId8

- Total quantity matched for SecId2 is 2700. (2600 + 100)

- Note: there are other combinations of matches among the orders which

would lead to the same result of 2700 total qty matching

- SecId3 has only one Buy order, no other orders to match against

- More Examples

- Example 1:

Orders in cache:

OrdId1 SecId1 Sell 100 User10 Company2

OrdId2 SecId3 Sell 200 User8 Company2

OrdId3 SecId1 Buy 300 User13 Company2

OrdId4 SecId2 Sell 400 User12 Company2

OrdId5 SecId3 Sell 500 User7 Company2

OrdId6 SecId3 Buy 600 User3 Company1

OrdId7 SecId1 Sell 700 User10 Company2

OrdId8 SecId1 Sell 800 User2 Company1

OrdId9 SecId2 Buy 900 User6 Cuompany2

OrdId10 SecId2 Sell 1000 User5 Company1

OrdId11 SecId1 Sell 1100 User13 Company2

OrdId12 SecId2 Buy 1200 User9 Company2

OrdId13 SecId1 Sell 1300 User1 Company

Total qty matching for security ids:

SecId1 300

SecId2 1000

SecId3 600

- Example 2:

Orders in cache:

OrdId1 SecId3 Sell 100 User1 Company1

OrdId2 SecId3 Sell 200 User3 Company2

OrdId3 SecId1 Buy 300 User2 Company1

OrdId4 SecId3 Sell 400 User5 Company2

OrdId5 SecId2 Sell 500 User2 Company1

OrdId6 SecId2 Buy 600 User3 Company2

OrdId7 SecId2 Sell 700 User1 Company1

OrdId8 SecId1 Sell 800 User2 Company1

OrdId9 SecId1 Buy 900 User5 Company2

OrdId10 SecId1 Sell 1000 User1 Company1

OrdId11 SecId2 Sell 1100 User6 Company2

Total qty matching for security ids:

SecId1 900

SecId2 600

SecId3 0