Part 1
Order book is a template class, with single template parameter denoting price quantization policy of the exchange. Two examples can be found in the header, including one toy example and one real life example for HKEX.

We need a two dimensional container for storing orders, one dimension for price level and one dimension for different orders within the same price level. The first dimension should be sorted by price level, the second dimension is sorted by queuing time (first come first serve based), besides efficient insersion and erasing should be supported in both dimensions, thus std::map<> and std::list<> are used as the containers for the first dimension and second dimension respectively.

Part 2
The performance of the operations are :

-	new order	searching time	= O(n)
		map's insertion time	= O(log(m))
		list's insertion time	= O(1)
-	amend order	searching time	= O(n)
		map's erasing time	= O(log(m))
		list's erasing time	= O(1)
-	cancel order	searching time	= O(n)
		map's erasing time	= O(log(m))
		list's erasing time	= O(1)
-	get price	O(1)	
-	get quantity	O(log(m)) + O(n)	

Remark : The dominant part in new_order, amend_order and cancel_order function is the searching time. Searching in new_order means the protection against duplicated order ID, while searching in amend_order and cancel_order means searching existing order book for the order with required ID. Since the orders are not sorted by order ID, hence the search is slow : O(n). There are alternatives that make searching faster, like : sorting order by order ID instead of price level and queuing time, yet this design will result in slower order book matching algorithms.

Part 3
To support real world applications, more attributes should be added to struct "Order" :
- order status, for example : new, partially filled, completely filled, rejected, cancelled
- order type, for example : limit order, market order, auction order
- time in force, for example : immediate or cancel, kill or fill
- maximum number of price levels that transaction can happen
- traded quantity or quantity left.

Avoid using double as price, instead, a new class "Price" should be implemented, so that price can be quantized into price level, which supports different manipulations such as addition, subtraction etc. Besides, order quantity should be unsigned.

OrderBook is a container of orders, we need iterator that allows fast iterating through order book (it can either be one dimensional iterator or two dimensional iterator), order's attributes can then be accessed or modified by dereferencing iterator. On top of OrderBook::iterator, different order book matching algorithms can be developed.