## Orderbook.h

```
#pragma once
```

---

```
#include <map>
#include <cmath>
#include <unordered_map>


struct Order
{
      int order_id;
      char side;
      int size;
      uint64_t price;
};

//Key for these maps = price*std::pow(10,8), this will keep
//comparisons consistent in the map
typedef std::map<uint64_t,Order,std::greater<uint64_t>> buySide_t;
typedef std::map<uint64_t,Order> sellSide_t;
typedef std::unordered_map<int,std::map<uint64_t,Order>::iterator> OrderIdMap_t;
typedef std::unordered_map<int,Order> OrderIdOrderMap_t;

class OrderBook
{
  private:
      sellSide_t sellSide;
      buySide_t buySide;
      Order orderDetails;
      OrderIdOrderMap_t orderIdOrderMap;
      OrderIdMap_t orderIdMap;

    public :

    OrderBook();
    ~OrderBook();
    void add(int order_id, char side, double price, int size);
    void modify(int order_id, int new_size);
    void remove(int order_id);
    double get_price(char side, int level);
    int get_size(char side, int level);
};
```

## orderbook.cpp

```
#include <string>
#include <iostream>
#include <iterator>
#include "orderbook.h"

OrderBook::OrderBook()
{
}

OrderBook::~OrderBook()
{
}


void OrderBook::add(int order_id, char side, double price, int size)
{
	try
	{
		orderDetails.order_id=order_id;
		orderDetails.side=side;
		orderDetails.price =price*std::pow(10,8);
		orderDetails.size=size;
		orderIdOrderMap.emplace(std::make_pair(order_id,orderDetails));

		if (side == 'S')
		{
			auto
result=sellSide.emplace(std::make_pair(orderDetails.price,orderDetails));
			if (result.second)
			{
				orderIdMap[order_id]=result.first;
			}
			else
			{
				result.first->second.size+=size;
			}
		}
		else
		{
			auto
result=buySide.emplace(std::make_pair(orderDetails.price,orderDetails));
			if (result.second)
			{
				orderIdMap[order_id]=result.first;
			}
			else
			{
				result.first->second.size+=size;
			}
		}

	}
	catch (...)
	{
		//relevent catches can be added
		std::cerr<<"Error Processing onAddOrder mesasage!!!!!\n";
	}
```

```
}

void OrderBook::modify(int order_id, int new_size)
{
	try
	{
		auto found = orderIdMap.find(order_id);
		auto foundOrderIdOrderMap = orderIdOrderMap.find(order_id);
		int oldSize=0;

		if ( foundOrderIdOrderMap == orderIdOrderMap.end() )
		{
/*			std::cout<<"Order id="<<order_id<<"\n";
			std::cerr<<"remove::!!!!Order not found in
orderIdOrderMap!!!!!!\n";*/
		}
		else
		{
			oldSize=(foundOrderIdOrderMap->second).size;
			(foundOrderIdOrderMap->second).size=new_size;
		}

		if ( found == orderIdMap.end() )
		{
			//std::cerr<<"Modify::!!!!Order not found in
orderIdMap!!!!!!\n";
		}
		else
		{
			auto sideMapit = found->second;
			(sideMapit->second).size=(sideMapit->second).size-
oldSize+new_size;
		}
	}
	catch (...)
	{//relevent catches can be added
		std::cerr<<"Error Processing modifyOrder mesasage!!!!!\n";
	}
}


void OrderBook::remove(int order_id)
{
	try
	{
		auto foundOrderIdMap = orderIdMap.find(order_id);
		auto foundOrderIdOrderMap = orderIdOrderMap.find(order_id);
		int delSize=0;

		if ( foundOrderIdOrderMap == orderIdOrderMap.end() )
		{
			//std::cout<<"Order id="<<order_id<<"\n";
			std::cerr<<"remove::!!!!Order not found in
orderIdOrderMap!!!!!!\n";
		}
		else
		{
			delSize=(foundOrderIdOrderMap->second).size;
			//std::cout<<"delsize:"<<delSize<<"\n";
			orderIdOrderMap.erase(foundOrderIdOrderMap);
```

```
		}


		/*for (auto &it : orderIdOrderMap)
			std::cout<< "OrderId="<<it.first<<"\n";

		for (auto &it2 : orderIdMap)
			std::cout<< "OrderId="<<it2.first<<"\n";*/

		if ( foundOrderIdMap == orderIdMap.end() )
		{
			//std::cout<<"Order id="<<order_id<<"\n";
			//std::cerr<<"remove::!!!!Order not found in
orderIdMap!!!!!!\n";
		}
		else
		{
			auto sideMapit = foundOrderIdMap->second;
			//std::cout<<"sizeInSideMap="<<(sideMapit->second).size<<"\n";
			(sideMapit->second).size -=delSize;
			if ((sideMapit->second).side == 'B')
			{
				if ((sideMapit->second).size ==0)
				{
					buySide.erase(sideMapit);
					orderIdMap.erase(foundOrderIdMap);
				}
			}
			else
			{
				//the order is in SELL side map
				if ((sideMapit->second).size ==0)
				{
					sellSide.erase(sideMapit);
					orderIdMap.erase(foundOrderIdMap);
				}
			}
		}
	}
	catch (...)
	{//relevent catches can be added
		std::cerr<<"Error Processing CancelledOrder mesasage!!!!!\n";
	}
}

double OrderBook::get_price(char side, int level)
{
	try
	{
		if (side == 'B')
		{
			if (!buySide.empty())
			{
				auto found= buySide.begin();
				if (level >1)
					std::advance(found,level-1);
				return (found->second.price)/std::pow(10,8);
			}
			else
				throw std::string("No Buy orders yet!!\n");
```

```
		}
		else
		{
			if (!sellSide.empty())
			{
				auto found= sellSide.begin();
				if (level >1)
					std::advance(found,level-1);
				return (found->second.price)/std::pow(10,8);
			}
			else
				throw std::string("No Sell orders yet!!\n");
		}

	}
	catch (std::string error)
	{
		std::cerr<<error;
	}
	catch (...)
	{//relevent catches can be added
		std::cerr<<"Error Processing CancelledOrder mesasage!!!!!\n";
	}
}

int OrderBook::get_size(char side, int level)
{
	try
	{
		if (side == 'B')
		{
			if (!buySide.empty())
			{
				auto found= buySide.begin();
				if (level >1)
					std::advance(found,level-1);
				return (found->second.size);
			}
			else
				throw std::string("No Buy orders yet!!\n");

		}
		else
		{
			if (!sellSide.empty())
			{
/*				for (auto it=sellSide.begin();it != sellSide.end();it++)
					std::cout<<it->second.price<<" ;"<<it-
>second.size<<"\n";*/
				auto found= sellSide.begin();
				if (level >1)
					std::advance(found,level-1);
				return (found->second.size);
			}
			else
				throw std::string("No Sell orders yet!!\n");
		}

	}
```

```
	catch (std::string error)
	{
		std::cerr<<error;
	}
	catch (...)
	{//relevent catches can be added
		std::cerr<<"Error Processing CancelledOrder mesasage!!!!!\n";
	}
}
```

## main.cpp

```cpp
#include <iostream>
#include "orderbook.h"

//const char *inputFile = "test.in";

int main(int argc, char **argv)
{

/*    Parser myParser(currentDate, "myTestFile");

    int fd = open(inputFile, O_RDONLY);
    if (fd == -1)
    {
        fprintf(stderr, "Couldn't open %s\n", inputFile);
        return 1;
    }



    close(fd);*/
      //###TestCase 1###
      OrderBook book;
      book.add(1,'B',45.2,100);
      book.modify(1,50);
      double price = book.get_price('B',1);
      std::cout<<price<<"\n";
      book.add( 2,'S',51.4,200);
      book.add(3,'B',45.1,100);
      int size = book.get_size ('S',1);
      std::cout<<size<<"\n";
      book.add(4,'S',51.2,300);
      book.add(5,'S',51.2,200);
      //book.add(6,'S',51,200);
      book.remove(3);
      price=book.get_price ('B',1);
      std::cout<<price<<"\n";
      size = book.get_size('B',1);
      std::cout<<size<<"\n";
      price=book.get_price( 'S',1);
      std::cout<<price<<"\n";
      //size = book.get_size('S',1);
      size = book.get_size('S',1);
      std::cout<<size<<"\n\n\n";


      //test case #2
      OrderBook book2;
      book2.add(1, 'B', 22.5, 100);
      book2.add( 2 ,'S', 37.8, 250);
      book2.add( 3 ,'B', 24.7, 150);
      price = book2.get_price( 'B', 1); //this returns 24.7
      std::cout<<price<<"\n";
      price = book2.get_price( 'B', 2 );//this returns 22.5
      std::cout<<price<<"\n";
      book2.modify( 3, 50);
      book2.add( 4, 'S', 35.1, 250);
      book2.add( 5, 'S', 37.8, 150);
      price=book2.get_price( 'S', 1);//this returns 35.1
      std::cout<<price<<"\n";
      book2.remove( 3);
```

```
        size=book2.get_size( 'S', 1 );//this returns 250
        std::cout<<size<<"\n";
        size=book2.get_size( 'S' ,2 );//this returns 400.
        std::cout<<size<<"\n";
        book2.remove( 5);
        book2.add( 6, 'S',37.8, 150);
        book2.add( 7, 'S', 37.6, 350);
        book2.add( 8, 'B', 24.7, 200);
        size=book2.get_size( 'B', 1 );//this returns 200
        std::cout<<size<<"\n";
        price=book2.get_price( 'S', 2);//this returns 37.6
        std::cout<<price<<"\n";
        book2.modify( 8 ,150);
        book2.add( 9 ,'S', 35.1, 200);
        book2.add( 10, 'B', 22.5, 350);
        size=book2.get_size( 'B', 2 );//this returns 450
        std::cout<<size<<"\n";
        price=book2.get_price( 'S', 3);//this returns 37.8
        std::cout<<price<<"\n";

    return 0;
}
```

## ReadMe.txt

**To compile:**

---

```
  'make'

To remove the binaries:
-----------------------
  'make clean'

To run the test case:
---------------------
type './orderbook' in the project directory


If more time is available the parser for the input file can be developed,
another point is
to use only 2 maps ; one with Map<price, qty> and another with
Map<OrderId,OrderDetails>
```

## Makefile:

```
---------
OBJS = Parser.o
```

```
all: orderbook

orderbook: main.cpp orderbook.cpp
	g++ -W -O3 -std=c++14 -o $@ $^

%.o : %.cc
	g++ -W -O3 -c -std=c++14 -o $@ $<

#libparser.a: $(OBJS)
#	ar rcs libparser.a $^

clean:
	rm -f *.o *.a feed
```