

Part I includes Test results and program source code

Test 1: deal with orders file

Input file name is orders.txt. Show the orders information.

```

141         bidBook.pop();
142         xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163 //if want to
164 int choice
165 do{
166     cout<<
167     cout<<
168     cout<<
169     cout<<"Enter 3 for view ask price order range: "<<endl;

```

Test 2: Output orders file running results.

Enter choice 1, the orders file output shows the bid/ask match results

```

stockOrderB
atch Price::28.86
Bid/Ask Orders execution::Time:14:09:28Ticker::TWTRType::SQuantity(Shares)::710M
atch Price::29.04
Bid/Ask Orders execution::Time:14:05:23Ticker::TWTRType::BQuantity(Shares)::190M
atch Price::29.77
Bid/Ask Orders execution::Time:13:08:11Ticker::TWTRType::BQuantity(Shares)::530M
atch Price::32.42
Bid/Ask Orders execution::Time:10:44:36Ticker::TWTRType::SQuantity(Shares)::970M
atch Price::30.11
Bid/Ask Orders execution::Time:15:31:22Ticker::TWTRType::BQuantity(Shares)::160M
atch Price::31.97
Bid/Ask Orders execution::Time:10:24:31Ticker::TWTRType::SQuantity(Shares)::880M
atch Price::30.11
Bid/Ask Orders execution::Time:14:22:34Ticker::TWTRType::SQuantity(Shares)::500M
atch Price::29.09
Bid/Ask Orders execution::Time:10:50:20Ticker::TWTRType::BQuantity(Shares)::400M
atch Price::29.77
=====Welcome to use stock order book simulation=====
=
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order <Time, Ticker, Type, Quantity, Price>:
Enter 5 for current top three orders <Price, Quantity>:
cout<<
cout<<"Enter 0 for exit order book simulation program: "<<endl;
cout<<"Enter 1 for looking at the filled/part filled orders: "<<endl;

```

Test 3: Output orders file running results.

Enter choice 2, shows the current unfilled bid orders price range

```

xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;
"C:\Program Files\CodeBlocks\share\CodeBlocks\projects\stockOrderBook\stockOrderBoo...
Enter 3 for view ask price order range:
Enter 4 for current top order <Time, Ticker, Type, Quantity, Price>:
Enter 5 for current top three orders <Price, Quantity>:
Enter 6 for input a new order <Time, Ticker, Type, Quantity, Price>:
Enter 7 for modify order <Type, Quantity, Price>:
Enter 8 for cancel order <Type, Price>:
2
Current Max bid order price: 29.47

Current Min bid order price: 27.64

=====Welcome to use stock order book simulation=====
=
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order <Time, Ticker, Type, Quantity, Price>:
Enter 5 for current top three orders <Price, Quantity>:
Enter 6 for input a new order <Time, Ticker, Type, Quantity, Price>:
Enter 7 for modify order <Type, Quantity, Price>:
Enter 8 for cancel order <Type, Price>:
=====
if want
int c
do{
    cout<<"Enter 3 for view ask price order range: "<<endl;
    cout<<"Enter 4 for current top order (Time, Ticker, Type, Quantity, Price): "<<endl;

```

Test 3: Output orders file running results.

Enter choice 3, shows the current unfilled ask orders price range

```

bidBook.pop();
xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;
"C:\Program Files\CodeBlocks\share\CodeBlocks\projects\stockOrderBook\stockOrderBoo...
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order <Time, Ticker, Type, Quantity, Price>:
Enter 5 for current top three orders <Price, Quantity>:
Enter 6 for input a new order <Time, Ticker, Type, Quantity, Price>:
Enter 7 for modify order <Type, Quantity, Price>:
Enter 8 for cancel order <Type, Price>:
3
Current Max ask order price: 32.91

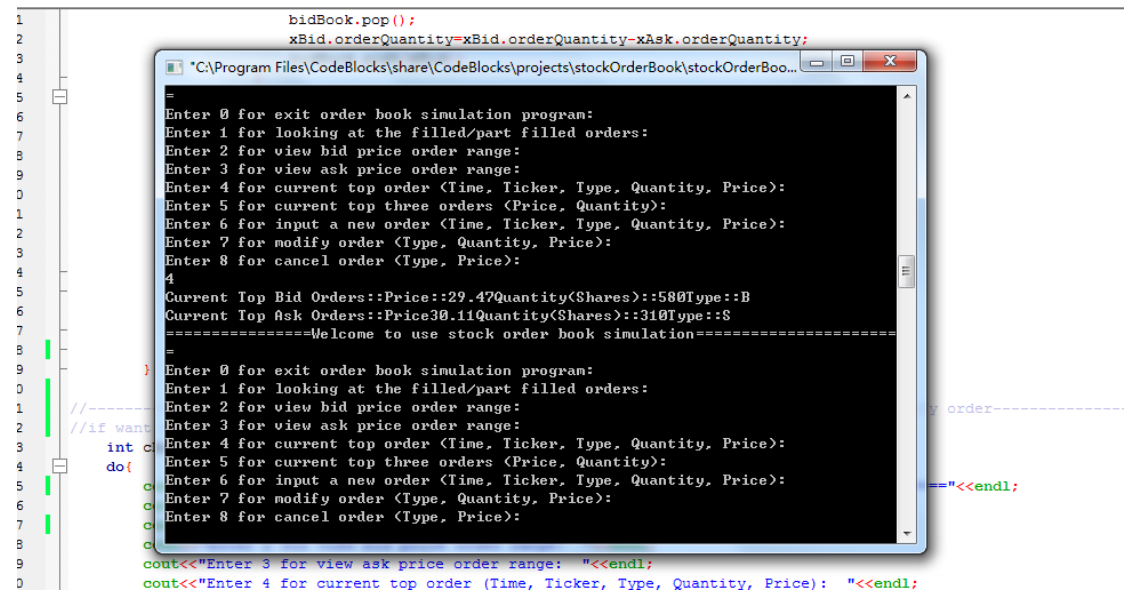
Current Min ask order price: 30.11

=====Welcome to use stock order book simulation=====
=
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order <Time, Ticker, Type, Quantity, Price>:
Enter 5 for current top three orders <Price, Quantity>:
Enter 6 for input a new order <Time, Ticker, Type, Quantity, Price>:
Enter 7 for modify order <Type, Quantity, Price>:
Enter 8 for cancel order <Type, Price>:
=====
if want
int c
do{
    cout<<"Enter 3 for view ask price order range: "<<endl;
    cout<<"Enter 4 for current top order (Time, Ticker, Type, Quantity, Price): "<<endl;

```

Test 4: Output orders file running results.

Enter choice 4, shows the current top bid/ask orders information



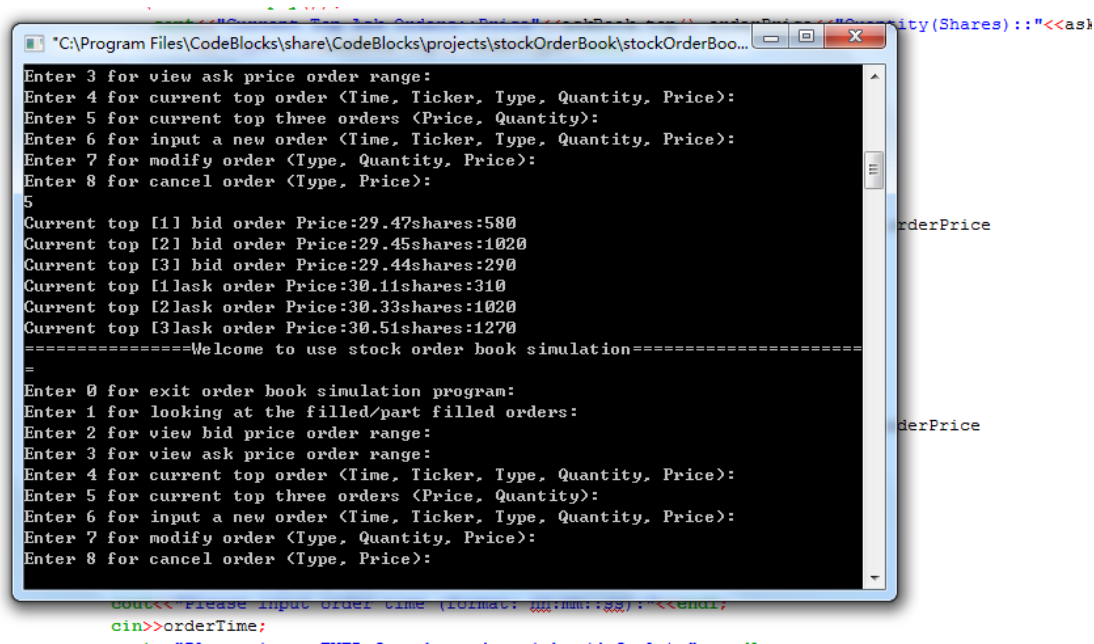
```

1      bidBook.pop();
2      xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;
3
4
5
6      Enter 0 for exit order book simulation program:
7      Enter 1 for looking at the filled/part filled orders:
8      Enter 2 for view bid price order range:
9      Enter 3 for view ask price order range:
10     Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
11     Enter 5 for current top three orders (Price, Quantity):
12     Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
13     Enter 7 for modify order (Type, Quantity, Price):
14     Enter 8 for cancel order (Type, Price):
15     4
16     Current Top Bid Orders::Price::29.47Quantity(Shares)::580Type::B
17     Current Top Ask Orders::Price30.11Quantity(Shares)::310Type::S
18     =====Welcome to use stock order book simulation=====
19     =
20     Enter 0 for exit order book simulation program:
21     Enter 1 for looking at the filled/part filled orders:
22     Enter 2 for view bid price order range:
23     Enter 3 for view ask price order range:
24     Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
25     Enter 5 for current top three orders (Price, Quantity):
26     Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
27     Enter 7 for modify order (Type, Quantity, Price):
28     Enter 8 for cancel order (Type, Price):
29
30     cout<<"Enter 3 for view ask price order range: "<<endl;
31     cout<<"Enter 4 for current top order (Time, Ticker, Type, Quantity, Price): "<<endl;

```

Test 5: Output orders file running results.

Enter choice 5, shows the current top 3 bid/ask orders information



```

1      bidBook.pop();
2      xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;
3
4
5
6      Enter 0 for exit order book simulation program:
7      Enter 1 for looking at the filled/part filled orders:
8      Enter 2 for view bid price order range:
9      Enter 3 for view ask price order range:
10     Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
11     Enter 5 for current top three orders (Price, Quantity):
12     Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
13     Enter 7 for modify order (Type, Quantity, Price):
14     Enter 8 for cancel order (Type, Price):
15     5
16     Current top [1] bid order Price:29.47shares:580
17     Current top [2] bid order Price:29.45shares:1020
18     Current top [3] bid order Price:29.44shares:290
19     Current top [1]ask order Price:30.11shares:310
20     Current top [2]ask order Price:30.33shares:1020
21     Current top [3]ask order Price:30.51shares:1270
22     =====Welcome to use stock order book simulation=====
23     =
24     Enter 0 for exit order book simulation program:
25     Enter 1 for looking at the filled/part filled orders:
26     Enter 2 for view bid price order range:
27     Enter 3 for view ask price order range:
28     Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
29     Enter 5 for current top three orders (Price, Quantity):
30     Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
31     Enter 7 for modify order (Type, Quantity, Price):
32     Enter 8 for cancel order (Type, Price):
33
34     cout<<"Please input order time (format: hh:mm:ss): "<<endl;
35     cin>>orderTime;

```

Test 6: Input new Order(bid)

Enter choice 6, shows the result.

Type: B, Quantity:520, Price:30.35, two bid/ask match, two ask orders executed.

```

*CA\Program Files\CodeBlocks\share\CodeBlocks\projects\stockOrderBook\stockOrderBook
Enter 3 for view ask price order range:
Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
Enter 5 for current top three orders (Price, Quantity):
Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
Enter 7 for modify order (Type, Quantity, Price):
Enter 8 for cancel order (Type, Price):
5
Current top [1] bid order Price:29.47shares:580
Current top [2] bid order Price:29.45shares:1020
Current top [3] bid order Price:29.44shares:290
Current top [1] ask order Price:30.11shares:310
Current top [2] ask order Price:30.33shares:1020
Current top [3] ask order Price:30.51shares:1270
=====Welcome to use stock order book simulation=====
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
Enter 5 for current top three orders (Price, Quantity):
Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
Enter 7 for modify order (Type, Quantity, Price):
Enter 8 for cancel order (Type, Price):

*CA\Program Files\CodeBlocks\share\CodeBlocks\projects\stockOrderBook\stockOrderBook
Please input a new order:
Please input order time (format: hh:mm:ss):
10:47:34
Please input TWTR for the order ticker(default):
TWTR
Please input order type(B or S, B for buy, S for sell):
B
Please input order quantity:
520
Please input order price:
30.35
New order execution:Time::10:47:34Ticker::TWTRType::BQuantity::210Price::30.3
New order execution:Time::14:06:34Ticker::TWTRType::SQuantity::310Price::30.1
=====Welcome to use stock order book simulation=====
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
Enter 5 for current top three orders (Price, Quantity):
Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
Enter 7 for modify order (Type, Quantity, Price):
Enter 8 for cancel order (Type, Price):
    
```

Test 7: Input new Order(ask)

Enter choice 6, shows the result.

Type:S, Quantity:450, Price:29.99, no bid/ask match

```

*CA\Program Files\CodeBlocks\share\CodeBlocks\projects\stockOrderBook\stockOrderBook
Enter 8 for cancel order (Type, Price):
6
Please input a new order:
Please input order time (format: hh:mm:ss):
11:23:45
Please input TWTR for the order ticker(default):
TWTR
Please input order type(B or S, B for buy, S for sell):
S
Please input order quantity:
450
Please input order price:
29.98
=====Welcome to use stock order book simulation=====
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
Enter 5 for current top three orders (Price, Quantity):
Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
Enter 7 for modify order (Type, Quantity, Price):
Enter 8 for cancel order (Type, Price):

else{//orderType=="S"
    askBook.push(v);
}
    
```

Test 8: Modify Order

Enter choice 7, shows the result.

Type:B, Quantity:180(old order:580), Price:29.47(current top bid order)

```

cout<<"Current top ["<<i<<"] bid order Price:"<<bidBookTmp2.top().orderPrice
//shares:"<<bidBookTmp2.top().orderQuantity<<endl;

Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
Enter 5 for current top three orders (Price, Quantity):
Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
Enter 7 for modify order (Type, Quantity, Price):
Enter 8 for cancel order (Type, Price):
7
Please input order type(B or S, B for buy, S for sell) that you want to modify :
B
Please input order price that you want to modify :
29.47
Please input new order quantity :
180
The original bid order:Time15:33:24Ticker::TWIRType::BQuantity(Shares)::580Price
::29.47
The modified bid order:Time15:33:24Ticker::TWIRType::BQuantity(Shares)::180Price
::29.47
=====Welcome to use stock order book simulation=====
=
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
else{//orderType=="S"

```

Test 9: Cancel Order

Enter choice 8, shows the result. Type:B, Price:29.47(current top bid order)

```

Enter 8 for cancel order (Type, Price):
5
Current top [1] bid order Price:29.47shares:580
Current top [2] bid order Price:29.45shares:1020
Current top [3] bid order Price:29.44shares:290
Current top [1]ask order Price:30.11shares:310
Current top [2]ask order Price:30.33shares:1020
Current top [3]ask order Price:30.51shares:1270
=====Welcome to use stock order book simulation=====
=
Enter 0 for exit order book simulation program:
Enter 1 for looking at the filled/part filled orders:
Enter 2 for view bid price order range:
Enter 3 for view ask price order range:
Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
Enter 5 for current top three orders (Price, Quantity):
Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
Enter 7 for modify order (Type, Quantity, Price):
Enter 8 for cancel order (Type, Price):
8
Please input order type(B or S, B for buy, S for sell) that you want to cancel :
B
Please input order price that you want to cancel :
29.47
Cancel the old bid order:Time15:33:24Ticker::TWIRType::BQuantity(Shares)::580Pri
ce::29.47
=====Welcome to use stock order book simulation=====
=
Enter 0 for exit order book simulation program:

```

```
//This program is to simulate stock transaction by implement a stock order book.  
//It will input "orders.txt" for orders file simulation and menu for viewing the results.  
//It also permit to input new orders, cancel orders or modify orders.  
//This program is free to use, you can redistribute it  
//and modify it under the software license.  
//Program Author: Zhen Qian (Martin), Rutgers University  
//Email:qianzhen77@hotmail.com, or zhen.qian@rutgers.edu.com  
//This program is distributed in the hope that it will be useful,  
//but without any warranty for a particular purpose.  
//This program is passed by Code::Blocks.  
// Copyright (c) Oct 2015
```

```
#include <iostream>  
#include <sstream>  
#include <fstream>  
#include <string>  
#include <queue>  
#include <stack>  
#include <vector>  
#include <algorithm>
```

```
using namespace std;
```

```
//define a class for bid/ask order format
```

```
struct order{  
    string orderTime;  
    string orderTicker;  
    string orderType;  
    int orderQuantity;  
    double orderPrice;
```

```
};
```

```
//define a class for bid orders comparison, great value gains priority,
```

```
//if prices equal, bid order with the early time stamp gains priority.
```

```
class compareBid{
```

```
public:
```

```
    bool operator()(order& p1, order& p2){
```

```
        if(p1.orderPrice<p2.orderPrice)
```

```
            return true;
```

```
        if(p1.orderPrice==p2.orderPrice && p1.orderTime<p2.orderTime)
```

```
            return true;
```

```
        return false;
```

```
    }
```

```
};
```

```
//define a class for ask orders comparison, small value gains priority,
```

```
//if prices equal, ask order with the early time stamp gains priority.
```

```
class compareAsk{
```

```
public:
```

```
    bool operator()(order& p1, order& p2){
```

```
        if(p1.orderPrice>p2.orderPrice)
```

```
            return true;
```

```
        if(p1.orderPrice==p2.orderPrice && p1.orderTime<p2.orderTime)
```

```
            return true;
```

```
        return false;
```

```
    }
```

```
};
```

```
//the main program is dividend with two parts.
```

//I: the first part is to deal with the orders by file, like the computer server in stock exchange.

//the huge bid/ask orders are automated to operate match function. The output is order execution results,current bid/ask top order

//II: the second part is to deal with the orders by screen input, like customs operation in client terminal.

//the customs can input new order to see the bid/ask order match if possible, or cancel/modify old bid/ask orders.

```
int main(){
```

```
//-----Part I: input bid/ask orders by file, like a server in stock exchange-----
```

```
//read the input order file, set the default file name:"orders.txt"
```

```
    ifstream infile;
```

```
    infile.open("orders.txt");
```

```
//declare order parameters
```

```
    string orderTime;
```

```
    string orderTicker;
```

```
    string orderType;
```

```
    int orderQuantity;
```

```
    double orderPrice;
```

```
//declare priority_queue container for building bid/ask book
```

```
//priority_queue's advantage, automatically, the top element of bid book has the highest bid price
```

```
//the top element of ask book has the lowest ask price
```

```
//if bid order price >= ask order price, bid-ask match
```

```
//if bid-ask match, saving filled/part filled orders in stack matchBook, last in first out (LIFO)
```

```
    priority_queue<order, vector<order>,compareBid > bidBook;
```

```
    priority_queue<order, vector<order>,compareAsk > askBook;
```



```
stack<order, vector<order> > matchBook;

//This program is only implementing one stock order book simulation, so the Ticker
is set as "TWTR" as default.

while(infile>>orderTime>>orderTicker>>orderType>>orderQuantity>>orderPrice){

    cout<<"Input Order:
Time::"<<orderTime<<"::Ticker"<<orderTicker<<"::Type"<<orderType<<"::Quant
ity"

    <<orderQuantity<<"::Price"<<orderPrice<<endl;

//Build a order object and every element in bid/ask priority_queue container has five
linked attributes

    order x={orderTime,orderTicker,orderType,orderQuantity,orderPrice};

//Buy side or Ask side orders

    if(orderType=="B"){

        bidBook.push(x);

//compare bid order price with ask order price

//if less than(<), leave it, if greater or equals(>=), execute bid/ask match function

//matchBook records one side order if it is disappeared, because the quantity(share)
is smaller.

//or records ask/bid orders both sides if bid order quantity equals ask order quantity.

        if(!askBook.empty()){

            while(bidBook.top().orderPrice>=askBook.top().orderPrice){

                if(bidBook.top().orderQuantity==askBook.top().orderQuantity){

                    order xBid=bidBook.top();

                    matchBook.push(xBid);

                    bidBook.pop();

                    order xAsk=askBook.top();

                    matchBook.push(xAsk);

                    askBook.pop();

                }
            }
        }
    }
}
```

```
else if(bidBook.top().orderQuantity>askBook.top().orderQuantity){
    order xAsk=askBook.top();
    matchBook.push(xAsk);
    askBook.pop();

    order xBid=bidBook.top();
    bidBook.pop();
    xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;
    bidBook.push(xBid);
}
else{ //bidBook.top().orderQuantity<askBook.top().orderQuantity
    order xBid=bidBook.top();
    matchBook.push(xBid);
    bidBook.pop();

    order xAsk=askBook.top();
    askBook.pop();
    xAsk.orderQuantity=xAsk.orderQuantity-xBid.orderQuantity;
    askBook.push(xAsk);
}
}
}
}

else if(orderType=="S"){
    askBook.push(x);
    if(!bidBook.empty()){
        while(bidBook.top().orderPrice>=askBook.top().orderPrice){
            if(bidBook.top().orderQuantity==askBook.top().orderQuantity){
                order xBid=bidBook.top();
                matchBook.push(xBid);
```

```
        bidBook.pop();

        order xAsk=askBook.top();
        matchBook.push(xAsk);
        askBook.pop();
    }
    else if(bidBook.top().orderQuantity>askBook.top().orderQuantity){
        order xAsk=askBook.top();
        matchBook.push(xAsk);
        askBook.pop();

        order xBid=bidBook.top();
        bidBook.pop();
        xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;
        bidBook.push(xBid);
    }
    else{ //bidBook.top().orderQuantity<askBook.top().orderQuantity
        order xBid=bidBook.top();
        matchBook.push(xBid);
        bidBook.pop();

        order xAsk=askBook.top();
        askBook.pop();
        xAsk.orderQuantity=xAsk.orderQuantity-xBid.orderQuantity;
        askBook.push(xAsk);
    }
}

}

}
```

```

    }

//-----Part II:input orders by screen, enter new order or
cancel/modify order-----

//if want to exit program, enter 0

    int choice;

    do{

        cout<<"=====Welcome to use stock order book
simulation===== "<<endl;

        cout<<"Enter 0 for exit order book simulation program: "<<endl;

        cout<<"Enter 1 for looking at the filled/part filled orders: "<<endl;

        cout<<"Enter 2 for view bid price order range: "<<endl;

        cout<<"Enter 3 for view ask price order range: "<<endl;

        cout<<"Enter 4 for current top order (Time, Ticker, Type, Quantity, Price):
"<<endl;

        cout<<"Enter 5 for current top three orders (Price, Quantity): "<<endl;

        cout<<"Enter 6 for input a new order (Time, Ticker, Type, Quantity, Price):
"<<endl;

//modify order only need to input three variables

        cout<<"Enter 7 for modify order (Type, Quantity, Price): "<<endl;

//cancel order only need to input two variables

        cout<<"Enter 8 for cancel order (Type, Price): "<<endl;


        cin>>choice;

        switch(choice){

            case 1:{

                while(!matchBook.empty()){

                    cout<<"Bid/Ask Orders
execution::Time:"<<matchBook.top().orderTime<<"Ticker::"<<matchBook.top().or
derTicker

                    <<"Type::"<<matchBook.top().orderType<<"Quantity(Shares)::"<<matchBook.top(
).orderQuantity<<"Match Price::"

```

```
<<matchBook.top().orderPrice<<endl;
matchBook.pop();
}
}break;
case 2:{
    priority_queue<order, vector<order>,compareBid > bidBookTmp;
    double maxBid;
    double minBid;
    double bidTmp;
    maxBid=bidBook.top().orderPrice;
    minBid=maxBid;
    bidBookTmp=bidBook;
    while(!bidBookTmp.empty()){
        bidTmp=bidBookTmp.top().orderPrice;
        if (bidTmp<minBid){
            minBid=bidTmp;
        }
        else{}
        bidBookTmp.pop();
    }
    cout<<"Current Max bid order price: "<<maxBid<<"\n"<<endl;;
    cout<<"Current Min bid order price: "<<minBid<<"\n"<<endl;
    }break;
case 3:{
    priority_queue<order, vector<order>,compareAsk > askBookTmp;
    double maxAsk;
    double minAsk;
    double askTmp;
    minAsk=askBook.top().orderPrice;
    maxAsk=minAsk;
```

```

askBookTmp=askBook;
while(!askBookTmp.empty()){
    askTmp=askBookTmp.top().orderPrice;
    if (askTmp>maxAsk){
        maxAsk=askTmp;
    }
    else{}
    askBookTmp.pop();
}

cout<<"Current Max ask order price: "<<maxAsk<<"\n"<<endl;
cout<<"Current Min ask order price: "<<minAsk<<"\n"<<endl;
}break;

case 4:{
    if (!bidBook.empty()){
        cout<<"Current Top Bid
Orders::Price::"<<bidBook.top().orderPrice<<"Quantity(Shares)::"<<bidBook.top().
orderQuantity
        <<"Type::"<<bidBook.top().orderType<<endl;
    }
    if (!askBook.empty()){
        cout<<"Current Top Ask
Orders::Price"<<askBook.top().orderPrice<<"Quantity(Shares)::"<<askBook.top().o
rderQuantity
        <<"Type::"<<askBook.top().orderType<<endl;
    }
    }break;

case 5:{
    priority_queue<order, vector<order>,compareBid > bidBookTmp2;
    bidBookTmp2=bidBook;

    for(int i=1;i<4;i++){
        cout<<"Current top ["<<i<<"] bid order
Price:"<<bidBookTmp2.top().orderPrice

```

```
        <<"shares:"<<bidBookTmp2.top().orderQuantity<<endl;
        bidBookTmp2.pop();
    }
    priority_queue<order, vector<order>,compareAsk > askBookTmp2;
    askBookTmp2=askBook;
    for(int j=1;j<4;j++){
        cout<<"Current top ["<<j<<"]ask order
Price:"<<askBookTmp2.top().orderPrice
        <<"shares:"<<askBookTmp2.top().orderQuantity<<endl;
        askBookTmp2.pop();
    }
    }break;
case 6:{
    cout<<"Please input a new order: "<<endl;
    cout<<"Please input order time (format: hh:mm:ss):"<<endl;
    cin>>orderTime;
    cout<<"Please input TWTR for the order ticker(default):"<<endl;
    cin>>orderTicker;
    cout<<"Please input order type(B or S, B for buy, S for sell):"<<endl;
    cin>>orderType;
    cout<<"please input order quantity:"<<endl;
    cin>>orderQuantity;
    cout<<"Please input order price:"<<endl;
    cin>>orderPrice;
    order x={orderTime,orderTicker,orderType,orderQuantity,orderPrice};
    if(orderType=="B"){
        bidBook.push(x);
    }
    else{//orderType=="S"
        askBook.push(x);
    }
}
```

```
}  
while (!matchBook.empty()){  
    matchBook.pop();  
}  
if(!askBook.empty()&&!bidBook.empty()){  
    while(bidBook.top().orderPrice>=askBook.top().orderPrice){  
        if(bidBook.top().orderQuantity==askBook.top().orderQuantity){  
            order xBid=bidBook.top();  
            matchBook.push(xBid);  
            bidBook.pop();  
  
            order xAsk=askBook.top();  
            matchBook.push(xAsk);  
            askBook.pop();  
        }  
        else if(bidBook.top().orderQuantity>askBook.top().orderQuantity){  
            order xAsk=askBook.top();  
            matchBook.push(xAsk);  
            askBook.pop();  
  
            order xBid=bidBook.top();  
            bidBook.pop();  
            xBid.orderQuantity=xBid.orderQuantity-xAsk.orderQuantity;  
            bidBook.push(xBid);  
        }  
        else{ //bidBook.top().orderQuantity<askBook.top().orderQuantity  
            order xBid=bidBook.top();  
            matchBook.push(xBid);  
            bidBook.pop();
```



```
        order xAsk=askBook.top();

        askBook.pop();

        xAsk.orderQuantity=xAsk.orderQuantity-xBid.orderQuantity;

        askBook.push(xAsk);

    }

}

}

while(!matchBook.empty()){

    cout<<"New order
execution::Time::"<<matchBook.top().orderTime<<"Ticker::"<<matchBook.top().or
derTicker

<<"Type::"<<matchBook.top().orderType<<"Quantity::"<<matchBook.top().orderQ
uantity<<"Price::"

        <<matchBook.top().orderPrice<<endl;

        matchBook.pop();

    }

    }break;

case 7:{

    cout<<"Please input order type(B or S, B for buy, S for sell) that you want
to modify :"<<endl;

    cin>>orderType;

    cout<<"Please input order price that you want to modify :"<<endl;

    cin>>orderPrice;

    cout<<"please input new order quantity :"<<endl;

    cin>>orderQuantity;

    if(orderType=="B"){

        priority_queue<order, vector<order>,compareBid > bidBookTmp3;
        priority_queue<order, vector<order>,compareBid > bidBookTmp4;
        bidBookTmp3=bidBook;

        order mBid=bidBookTmp3.top();
```

```
while(!bidBookTmp3.empty()){

    if(orderPrice==bidBookTmp3.top().orderPrice){

        cout<<"The original bid
order:Time"<<bidBookTmp3.top().orderTime<<"Ticker::"<<bidBookTmp3.top().ord
erTicker

        <<"Type::"<<bidBookTmp3.top().orderType<<"Quantity(Shares)::"<<bidBookTmp
3.top().orderQuantity<<"Price::"

        <<bidBookTmp3.top().orderPrice<<endl;

        cout<<"The modified bid
order:Time"<<bidBookTmp3.top().orderTime<<"Ticker::"<<bidBookTmp3.top().ord
erTicker

        <<"Type::"<<bidBookTmp3.top().orderType<<"Quantity(Shares)::"<<orderQuantit
y<<"Price::"

        <<bidBookTmp3.top().orderPrice<<endl;

        order
mBid={bidBookTmp3.top().orderTime,bidBookTmp3.top().orderTicker,
bidBookTmp3.top().orderType,

            orderQuantity, bidBookTmp3.top().orderPrice};

        bidBookTmp4.push(mBid);

        bidBookTmp3.pop();

    }

    else{

        bidBookTmp4.push(mBid);

        bidBookTmp3.pop();

    }

    swap(bidBook,bidBookTmp4);

}

else{//orderType=="B"

    priority_queue<order, vector<order>,compareAsk > askBookTmp3;

    priority_queue<order, vector<order>,compareAsk > askBookTmp4;

    askBookTmp3=askBook;
```

```
        order mAsk=askBookTmp3.top();

        while(!askBookTmp3.empty()){

            if(orderPrice==askBookTmp3.top().orderPrice){

                cout<<"The original bid
order:Time"<<askBookTmp3.top().orderTime<<"Ticker::"<<askBookTmp3.top().ord
erTicker

                <<"Type::"<<askBookTmp3.top().orderType<<"Quantity(Shares)::"<<askBookTmp
3.top().orderQuantity<<"Price::"

                <<askBookTmp3.top().orderPrice<<endl;

                cout<<"The modified bid
order:Time"<<askBookTmp3.top().orderTime<<"Ticker::"<<askBookTmp3.top().ord
erTicker

                <<"Type::"<<askBookTmp3.top().orderType<<"Quantity(Shares)::"<<orderQuantit
y<<"Price::"

                <<askBookTmp3.top().orderPrice<<endl;

                order
mAsk={askBookTmp3.top().orderTime,askBookTmp3.top().orderTicker,
askBookTmp3.top().orderType,

                orderQuantity, askBookTmp3.top().orderPrice};

                askBookTmp4.push(mAsk);

                askBookTmp3.pop();

            }

            else{

                askBookTmp4.push(mAsk);

                askBookTmp3.pop();

            }

        }

        swap(askBook,askBookTmp4);

    }

    }break;

case 8:{
```

```
    cout<<"Please input order type(B or S, B for buy, S for sell) that you want  
to cancel :"<<endl;  
  
    cin>>orderType;  
  
    cout<<"Please input order price that you want to cancel :"<<endl;  
  
    cin>>orderPrice;  
  
    if(orderType=="B"){  
        priority_queue<order, vector<order>,compareBid > bidBookTmp5;  
        priority_queue<order, vector<order>,compareBid > bidBookTmp6;  
        bidBookTmp5=bidBook;  
        order mBid=bidBookTmp5.top();  
        if(orderPrice==bidBookTmp5.top().orderPrice){  
            cout<<"Cancel the old bid  
order:Time"<<bidBookTmp5.top().orderTime<<"Ticker::"<<bidBookTmp5.top().ord  
erTicker  
  
            <<"Type::"<<bidBookTmp5.top().orderType<<"Quantity(Shares)::"<<bidBookTmp  
5.top().orderQuantity  
                <<"Price::"<<bidBookTmp5.top().orderPrice<<endl;  
            bidBookTmp5.pop();  
        }  
        else{  
            bidBookTmp6.push(mBid);  
            bidBookTmp5.pop();  
        }  
        swap(bidBook,bidBookTmp6);  
    }  
    else{//orderType=="S"  
        priority_queue<order, vector<order>,compareAsk > askBookTmp5;  
        priority_queue<order, vector<order>,compareAsk > askBookTmp6;  
        askBookTmp5=askBook;  
        order mAsk=askBookTmp5.top();
```

```
        if(orderPrice==askBookTmp5.top().orderPrice){
            cout<<"Cancel the old ask
order:"<<askBookTmp5.top().orderTime<<"::"<<askBookTmp5.top().orderTicker
<<"::"<<askBookTmp5.top().orderType<<"::"<<askBookTmp5.top().orderQuantity
<<"::"
            <<askBookTmp5.top().orderPrice<<endl;
            askBookTmp5.pop();
        }
        else{
            askBookTmp6.push(mAsk);
            askBookTmp5.pop();
        }
        swap(askBook,askBookTmp6);
    }
    }break;
}
}while(choice!=0);
return 0;
}
```