Ve 281 Project 4

Dong Jing
515370910182

main.h

```
#ifndef PROJECT_4_MARKET_H
#define PROJECT_4_MARKET_H

struct client{
    int time, type, price, quantity, duration, order;
    std::string name, symbol;
};

struct compare{
    bool operator()(client a, client b)
    {
        if (a.type==0)//buy
            if (a.price<b.price) return true;
            else if (a.price>b.price) return false;
            else return a.order>b.order;
        else if (a.price>b.price) return true;
            else if (a.price<b.price) return false;
            else return a.order>b.order;
    }
};
struct transfer{
    int number_buy, number_sell, net;
};

struct traveler{
    std::string symbol;
    int buy_time, sell_time, buy_price, sell_price, order, lowest_time,
lowest_price;
};

typedef std::priority_queue<client, std::vector<client>, compare>
client_heap;
typedef std::map<std::string, client_heap> type_map;
typedef std::map<std::string, std::priority_queue<int> > trade_max;
typedef std::map<std::string, std::priority_queue<int, std::vector<int>,
std::greater<int>>> trade_min;
typedef std::map<std::string, transfer> map_transfer;
typedef std::map<int, traveler> map_traveler;
typedef std::map<std::string, traveler> map_string_traveler;
```

```
#endif // PROJECT_4_MARKET_H
```

main.cpp

```
#include <iostream>
#include <unordered_map>
#include <string>
#include <sstream>
#include <queue>
#include <map>
#include <getopt.h>
#include <ctime>
#include <set>
#include "main.h"

using namespace std;

void Median(trade_max &m1, trade_min &m2, int current_time)
{
    auto it1=m1.begin();
    auto it2=m2.begin();
    int median=-1;
    while (it1!=m1.end())
    {
        median=-1;
        if ((!it1->second.empty())||(!it2->second.empty()))
        if (it1->second.size()==it2->second.size())
            median=(it1->second.top()+it2->second.top())/2;
        else if (it1->second.size()<it2->second.size())
            median=it2->second.top();
        else
            median=it1->second.top();
        if (median!=-1)
            cout<<"Median match price of "<<it1->first<<" at time
"<<current_time<<" is $"<<median<<endl;
        it1++;
        it2++;
    }
}

void Midpoint(type_map &buy, type_map &sell, int current_time)
{
    auto it1=buy.begin();
    auto it2=sell.begin();
    int midpoint=-1;
    while (it1!=buy.end())
    {
        while (!it1->second.empty())
        {
            if (it1->second.top().duration==-1) break;
            if
(it1->second.top().time+it1->second.top().duration>current_time) break;
            it1->second.pop();
        }
        while (!it2->second.empty())
        {
            if (it2->second.top().duration==-1) break;
            if
(it2->second.top().time+it2->second.top().duration>current_time) break;
            it2->second.pop();
        }
        if (!it1->second.empty() && !it2->second.empty())
            cout<<"Midpoint of "<<it1->first<<" at time "<<current_time<<"
is $"<<(it1->second.top().price+it2->second.top().price)/2<<endl;
        else
            cout<<"Midpoint of "<<it1->first<<" at time "<<current_time<<"
is undefined\n";
        it1++;
        it2++;
    }
}

int main(int argc, char *argv[])
{
    char opt;
    char *short_opts=(char *)"vmptg:";
    struct option long_opts[]={
        {"verbose", no_argument, NULL, 'v'},
        {"median", no_argument, NULL, 'm'},
        {"midpoint", no_argument, NULL, 'p'},
        {"transfers", no_argument, NULL, 't'},
        {"ttt", required_argument, NULL, 'g'},
        {0, 0, 0, 0}
        };
    int fee, num_shares, num_trades, amount;
    fee=0; amount=0; num_trades=0; num_shares=0;
    bool verbose,median,midpoint,transfers,ttt;
    verbose=false;
    median=false;
```

```
midpoint=false;
transfers=false;
ttt=false;
int n=1;
int price,quantity;
traveler temp;
temp.buy_time=-1;
temp.sell_time=-1;
temp.buy_price=-1;
temp.sell_price=-1;
temp.lowest_time=-1;
temp.lowest_price=-1;
map_traveler t;
map_string_traveler t2;
while ((opt=getopt_long(argc,argv,short_opts,long_opts,NULL))!=-1)
{
    switch (opt)
    {
        case 'v': verbose=true;break;
        case 'm': median=true;break;
        case 'p': midpoint=true;break;
        case 't': transfers=true;break;
        case 'g':
            {
                ttt=true;
                temp.symbol=optarg;
                temp.order=n;
                t.insert(make_pair(n,temp));
                t2.insert(make_pair(optarg,temp));
                n++;
                break;
            }
        default: cout<<"Wrong Arguments!\n";exit(0);
    }
}
string str;
istringstream iStream;
string goal;
int duration;
char c;
client tem;
client tem1;
int current_time=0;
trade_max m1;
```

```
    trade_max::iterator it_max;
    priority_queue<int> no_max;
    trade_min m2;
    trade_min::iterator it_min;
    priority_queue<int, vector<int>, greater<int>> no_min;
    type_map Buy;
    type_map Sell;
    client_heap none;
    map_transfer trans;
    type_map::iterator it;
    map_transfer::iterator it_trans;
    transfer notrans;
    notrans.number_sell=0;
    notrans.number_buy=0;
    notrans.net=0;
    int order=0;
    map_traveler::iterator it_traveler;
    map_string_traveler::iterator it_st;
    while (getline(cin,str))
    {
        if (str=="exit") break;
        iStream.str(str);

iStream>>tem.time>>tem.name>>goal>>tem.symbol>>c>>tem.price>>c>>tem.quantit
y>>tem.duration;
        iStream.clear();
        tem.order=order;
        order++;
        if (trans.find(tem.name)==trans.end())
            trans.insert(make_pair(tem.name,notrans));
        if (goal=="BUY") tem.type=0;
        else tem.type=1;
        if (median)
        {
            if (m1.find(tem.symbol)==m1.end())
                m1.insert(make_pair(tem.symbol,no_max));
            if (m2.find(tem.symbol)==m2.end())
                m2.insert(make_pair(tem.symbol,no_min));
        }
        if (tem.time!=current_time)
        {
            if (median) Median(m1,m2,current_time);
            if (midpoint) Midpoint(Buy,Sell,current_time);
            current_time=tem.time;
        }
        if (tem.type==0)//buy
        {
            if (ttt)
            {
                it_st=t2.find(tem.symbol);
        if (it_st!=t2.end())
        {
            if ((it_st->second.buy_time!=-1) &&
(tem.price>it_st->second.sell_price))
            {
            it_st->second.sell_price=tem.price;
            it_st->second.sell_time=current_time;
            t[it_st->second.order]=it_st->second;
            }
            if ((it_st->second.lowest_time!=-1) && (tem.price-
it_st->second.lowest_price>it_st->second.sell_price-
it_st->second.buy_price))
            {
            it_st->second.sell_price=tem.price;
            it_st->second.sell_time=current_time;
            it_st->second.buy_time=it_st->second.lowest_time;
            it_st->second.buy_price=it_st->second.lowest_price;
            it_st->second.lowest_price=-1;
            it_st->second.lowest_time=-1;
            t[it_st->second.order]=it_st->second;
            }
        }
            }
            if (Buy.find(tem.symbol)==Buy.end())
            {
                Buy.insert(make_pair(tem.symbol, none));
                Sell.insert(make_pair(tem.symbol, none));
        it=Buy.find(tem.symbol);
        if (tem.duration!=0)
                it->second.push(tem);
            }
            else
            {
                it=Sell.find(tem.symbol);
                if (!it->second.empty())
                {
                    while (it->second.top().price<=tem.price)
                    {
                        if
((it->second.top().time+it->second.top().duration<=current_time)&&(it->seco
nd.top().duration!=-1))
                        {
                            it->second.pop();
                        }
                        else
                        {
                        if (it->second.top().quantity>=tem.quantity)
                        {
                            if (verbose)
                            {
                                cout<<tem.name<<" purchased
"<<tem.quantity<<" shares of "<<tem.symbol<<" from
"<<it->second.top().name<<" for $"<<it->second.top().price<<"/share\n";
                            }

fee=fee+tem.quantity*it->second.top().price/100*2;

amount=amount+tem.quantity*it->second.top().price;
                            num_trades=num_trades+1;
                            num_shares=num_shares+tem.quantity;
                            tem1=it->second.top();
                            it->second.pop();
                            tem1.quantity=tem1.quantity-tem.quantity;
                price=tem1.price;
                    quantity=tem.quantity;
                            if (tem1.quantity>=0) it->second.push(tem1);
                            it_trans=trans.find(it->second.top().name);

it_trans->second.number_sell=it_trans->second.number_sell+quantity;

it_trans->second.net=it_trans->second.net+price*quantity;
                            it_trans=trans.find(tem.name);

it_trans->second.number_buy=it_trans->second.number_buy+quantity;
                            it_trans->second.net=it_trans->second.net-
price*quantity;
                            tem.quantity=0;
                            if (median)
                            {
                                it_max=m1.find(tem.symbol);
                                it_min=m2.find(tem.symbol);
                                if
(it_max->second.empty()||it_max->second.top()>=price)
it_max->second.push(price);
                                else it_min->second.push(price);
                                while
(it_max->second.size()>it_min->second.size()+1)
                                {

it_min->second.push(it_max->second.top());
                                    it_max->second.pop();
                                }
                                while
(it_min->second.size()>it_max->second.size()+1)
                                {

it_max->second.push(it_min->second.top());
                                    it_min->second.pop();
                                }
                            }
                            if (it->second.top().quantity==0)
it->second.pop();
                            break;
                        }
                        else
                        {
                            if (verbose)
                            {
                                cout<<tem.name<<" purchased
"<<it->second.top().quantity<<" shares of "<<tem.symbol<<" from
"<<it->second.top().name<<" for $"<<it->second.top().price<<"/share\n";
                            }

fee=fee+it->second.top().quantity*it->second.top().price/100*2;

amount=amount+it->second.top().quantity*it->second.top().price;
                            num_trades=num_trades+1;

num_shares=num_shares+it->second.top().quantity;
                price=it->second.top().price;
                quantity=it->second.top().quantity;
                            it_trans=trans.find(it->second.top().name);

it_trans->second.number_sell=it_trans->second.number_sell+quantity;

it_trans->second.net=it_trans->second.net+price*quantity;
```

```
                            it_trans=trans.find(tem.name);

it_trans->second.number_buy=it_trans->second.number_buy+quantity;
                            it_trans->second.net=it_trans->second.net-
price*quantity;
                            tem.quantity=tem.quantity-quantity;
                            if (median)
                            {
                                it_max=m1.find(tem.symbol);
                                it_min=m2.find(tem.symbol);
                                if
(it_max->second.empty()||it_max->second.top()>=price)
it_max->second.push(price);
                                else it_min->second.push(price);
                                while
(it_max->second.size()>it_min->second.size()+1)
                                {

it_min->second.push(it_max->second.top());
                                    it_max->second.pop();
                                }
                                while
(it_min->second.size()>it_max->second.size()+1)
                                {

it_max->second.push(it_min->second.top());
                                    it_min->second.pop();
                                }
                            }
                            it->second.pop();
                        }
                        }
                        if (it->second.empty()) break;
                    }
                    if ((tem.quantity>0)&&(tem.duration!=0))
                    {
                        it=Buy.find(tem.symbol);
                        it->second.push(tem);
                    }
                }
                else
                {
                    if (tem.duration!=0)
                    {
                        it=Buy.find(tem.symbol);
                        it->second.push(tem);
                    }
                }
            }
        }
        else //sell
        {
            if (ttt)
            {
        it_st=t2.find(tem.symbol);
                if (it_st!=t2.end())
        {
            if (it_st->second.buy_time==-1)
            {
            it_st->second.buy_price=tem.price;
            it_st->second.buy_time=current_time;
            t[it_st->second.order]=it_st->second;
            }
            else
            {
            if ((tem.price<it_st->second.buy_price) &&
(it_st->second.sell_price==-1))
            {
                it_st->second.buy_price=tem.price;
                it_st->second.buy_time=current_time;
                t[it_st->second.order]=it_st->second;
            }
            if ((tem.price<it_st->second.buy_price) &&
((it_st->second.lowest_price==-1)||(it_st->second.lowest_price>tem.price)))
            {
                it_st->second.lowest_price=tem.price;
                it_st->second.lowest_time=current_time;
                t[it_st->second.order]=it_st->second;
            }
            }
        }
            }
            if (Sell.find(tem.symbol)==Sell.end())
            {
                Buy.insert(make_pair(tem.symbol, none));
                Sell.insert(make_pair(tem.symbol, none));
        it=Sell.find(tem.symbol);
        if (tem.duration!=0);
```

```
                it->second.push(tem);
            }
            else
            {
                it=Buy.find(tem.symbol);
                if (!it->second.empty())
                {
                    while (it->second.top().price>=tem.price)
                    {
                        if ((it->second.top().duration!=-
1)&&(it->second.top().time+it->second.top().duration<=current_time))
                            it->second.pop();
                        else
                        {
                        if (it->second.top().quantity>=tem.quantity)
                        {
                            if (verbose)
                            {
                                cout<<it->second.top().name<<" purchased
"<<tem.quantity<<" shares of "<<tem.symbol<<" from "<<tem.name<<" for
$"<<it->second.top().price<<"/share\n";
                            }
                price=it->second.top().price;
                quantity=tem.quantity;

fee=fee+tem.quantity*it->second.top().price/100*2;

amount=amount+tem.quantity*it->second.top().price;
                            num_trades++;
                            num_shares=num_shares+tem.quantity;
                            tem1=it->second.top();
                            it->second.pop();
                            tem1.quantity=tem1.quantity-tem.quantity;
                            if (tem1.quantity>=0) it->second.push(tem1);
                            it_trans=trans.find(it->second.top().name);

it_trans->second.number_buy=it_trans->second.number_buy+quantity;
                            it_trans->second.net=it_trans->second.net-
price*quantity;
                            it_trans=trans.find(tem.name);

it_trans->second.number_sell=it_trans->second.number_sell+quantity;

it_trans->second.net=it_trans->second.net+price*quantity;
```

```
                              tem.quantity=0;
                              if (median)
                              {
                                  it_max=m1.find(tem.symbol);
                                  it_min=m2.find(tem.symbol);
                                  if
(it_max->second.empty()||it_max->second.top()>=price)
it_max->second.push(price);
                                  else it_min->second.push(price);
                                  while
(it_max->second.size()>it_min->second.size()+1)
                                  {

it_min->second.push(it_max->second.top());
                                      it_max->second.pop();
                                  }
                                  while
(it_min->second.size()>it_max->second.size()+1)
                                  {

it_max->second.push(it_min->second.top());
                                      it_min->second.pop();
                                  }
                              }
                              if (it->second.top().quantity==0)
it->second.pop();
                              break;
                          }
                          else
                          {
                              if (verbose)
                              {
                                  cout<<it->second.top().name<<" purchased
"<<it->second.top().quantity<<" shares of "<<tem.symbol<<" from
"<<tem.name<<" for $"<<it->second.top().price<<"/share\n";
                              }
                price=it->second.top().price;
                quantity=it->second.top().quantity;

fee=fee+it->second.top().quantity*it->second.top().price/100*2;

amount=amount+it->second.top().quantity*it->second.top().price;
                              num_trades++;
```

```
num_shares=num_shares+it->second.top().quantity;
                            it_trans=trans.find(it->second.top().name);

it_trans->second.number_buy=it_trans->second.number_buy+quantity;
                            it_trans->second.net=it_trans->second.net-
price*quantity;
                            it_trans=trans.find(tem.name);

it_trans->second.number_sell=it_trans->second.number_sell+quantity;

it_trans->second.net=it_trans->second.net+price*quantity;
                            tem.quantity=tem.quantity-quantity;
                            if (median)
                            {
                                it_max=m1.find(tem.symbol);
                                it_min=m2.find(tem.symbol);
                                if
(it_max->second.empty()||it_max->second.top()>=price)
it_max->second.push(price);
                                else it_min->second.push(price);
                                while
(it_max->second.size()>it_min->second.size()+1)
                                {

it_min->second.push(it_max->second.top());
                                    it_max->second.pop();
                                }
                                while
(it_min->second.size()>it_max->second.size()+1)
                                {

it_max->second.push(it_min->second.top());
                                    it_min->second.pop();
                                }
                            }
                            it->second.pop();
                        }
                        }
                        if (it->second.empty()) break;
                    }
                    if ((tem.quantity>0)&&(tem.duration!=0))
                    {
                        it=Sell.find(tem.symbol);
                        it->second.push(tem);
                    }
                }
                else
                {
                    if (tem.duration!=0)
                    {
                        it=Sell.find(tem.symbol);
                        it->second.push(tem);
                    }
                }
            }
        }
    }
    if (median) Median(m1,m2,current_time);
    if (midpoint) Midpoint(Buy,Sell,current_time);
    cout<<"---End of Day---\n";
    cout<<"Commission Earnings: $"<<fee<<endl;
    cout<<"Total Amount of Money Transferred: $"<<amount<<endl;
    cout<<"Number of Completed Trades: "<<num_trades<<endl;
    cout<<"Number of Shares Traded: "<<num_shares<<endl;
    if (transfers)
    {
        it_trans=trans.begin();
        while (it_trans!=trans.end())
        {
            cout<<it_trans->first<<" bought
"<<it_trans->second.number_buy<<" and sold
"<<it_trans->second.number_sell<<" for a net transfer of
$"<<it_trans->second.net<<endl;
            it_trans++;
        }
    }
    if (ttt)
    {
        for (int i=1;i<n;i++)
        {
        if ((t[i].buy_time!=-1)&&(t[i].sell_time!=-1))
            cout<<"Time travelers would buy "<<t[i].symbol<<" at time:
"<<t[i].buy_time<<" and sell it at time: "<<t[i].sell_time<<endl;
        else
        cout<<"Time travelers would buy "<<t[i].symbol<<" at time: "<<-1<<"
and sell it at time: "<<-1<<endl;
        }
    }
```

```
}
```

Makefile

```
all: main

main: main.o
    g++ -o main main.o

main.o: main.cpp
    g++ -c main.cpp -std=c++11

clean: rm -f main*.o
```