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

static const struct option long_options[] = {
    {"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'}
};
int main(int argc, char *argv[]){
    std::ios::sync_with_stdio(false);
    std::cin.tie(0);
    int opt;
    bool median=false,midpoint=false,transfer=false,verbose=false;
    std::vector<std::string> tttnames;
    while ((opt=getopt_long(argc,argv,"vmptg:",long_options,NULL))!=EOF){
        switch(opt){
            case 'v':
                verbose = true;
                break;
            case 'm':
                median = true;
                break;
            case 'p':
                midpoint = true;
                break;
            case 't':
                transfer = true;
                break;
            case 'g':
                tttnames.emplace_back(optarg);
                break;
            default:
                std::cout << "Invalid argument" << std::endl;
        }
    }
    orderbook ob(midpoint,median,transfer,verbose,tttnames);
    int ts = 0;
    int oid = 0;
    while (std::cin){
        std::string tmp;
        std::getline(std::cin,tmp);
        if (tmp.empty()) break;
        order curr_order = ob.order_generate(oid,tmp);
        oid++;
        ob.order_execute(curr_order);
    }
    ob.endoftime();
    ob.endofday();
    return 0;
}
```

main.cpp

```cpp
#ifndef EQUITY_H
```

```
#define EQUITY_H
#include "order.h"
#include <queue>
#include <string>
#include <iostream>
#include <limits>
class equity{
    int eid;
    std::string ename;
    std::priority_queue<order> buyers,sellers;
    std::vector<int> dealt;
    int median;
    int btime,stime,maxv,mintime,minb;
    public:
        equity(int eid,std::string ename):eid(eid),ename(ename),buyers(),sellers(),
            dealt(),median(-1),btime(-1),stime(-1),mintime(-1),minb(INT32_MAX),maxv(
            INT32_MIN){}
        void addbuyer(order v){buyers.push(v);}
        void addseller(order v){sellers.push(v);}
        order getbuyer();
        order getseller();
        int sellercount(){return sellers.size();}
        int buyercount(){return buyers.size();}
        void adddealtprice(int p);
        int getmedian(){return median;}
        void modify(int ts,int val,bool buy);
        void strategyout(){
            std::cout << "Time travelers would buy " << ename<< " at time: "<< btime
                <<" and sell it at time: "<< stime << std::endl;
        }
        std::string getname(){return ename;}
};
#endif
```

equity.h

```
#include "equity.h"
#include <algorithm>
order equity::getbuyer()
{
    order v = buyers.top();
    buyers.pop();
    return v;
}

order equity::getseller()
{
    order v = sellers.top();
    sellers.pop();
    return v;
}

void equity::modify(int ts,int val, bool buy){
    if (!buy){
            if (val<minb){
                mintime = ts;
                minb = val;
            }
```

```
    }
    else {
            if (mintime!=-1&&val-minb > maxv){
                btime = mintime;
                stime = ts;
                maxv = val-minb;

    }
    }
}
void equity::adddealtprice(int p){
    dealt.push_back(p);
    int a = dealt.size()/2;
    int b = (dealt.size()-1)/2;
    std::nth_element(dealt.begin(),dealt.begin()+a,dealt.end());
    int aa= dealt[a];
    std::nth_element(dealt.begin(),dealt.begin()+b,dealt.end());
    int bb = dealt[b];
    this->median = (aa+bb)/2;
}
```

equity.cpp

```
#ifndef CLIENT_H
#define CLIENT_H
#include <string>
class client{
    int cid;
    int income;
    std::string cname;
    int bought,sold;
    public:
        client(int cid,std::string cname,int income):cid(cid),cname(cname),income(
            income),bought(0),sold(0){}
        int getcid(){return cid;}
        int getincome(){return income;}
        int buy(int share,int val){bought+=share;income-=val;}
        int sell(int share,int val){sold+=share;income+=val;}
        std::string getname(){return cname;}
        int getbought(){return bought;}
        int getsold(){return sold;}
};
#endif
```

client.h

```
#ifndef ORDER_H
#define ORDER_H
class order{
    int oid;
    int cid,eid;
    bool buy;
    int share;
    int duration,ts;
    int price;
    public:
        friend bool operator<(const order &a,const order &b);
```

```
        order(int oid,int cid,int eid,bool buy,int share,int duration,int ts,int
            price):
            oid(oid),cid(cid),eid(eid),buy(buy),share(share),duration(duration),ts(
                ts),price(price){}
        void doneshare(int val){share-=val;}
        int getoid(){return oid;}
        int getcid(){return cid;}
        int geteid(){return eid;}
        bool getbuy(){return buy;}
        int getduration(){return duration;}
        int getprice(){return price;}
        int getshare(){return share;}
        int getts(){return ts;}
};

bool operator<(const order &a,const order &b);
#endif
```

order.h

```
#ifndef ORDER_H
#define ORDER_H
class order{
    int oid;
    int cid,eid;
    bool buy;
    int share;
    int duration,ts;
    int price;
    public:
        friend bool operator<(const order &a,const order &b);
        order(int oid,int cid,int eid,bool buy,int share,int duration,int ts,int
            price):
            oid(oid),cid(cid),eid(eid),buy(buy),share(share),duration(duration),ts(
                ts),price(price){}
        void doneshare(int val){share-=val;}
        int getoid(){return oid;}
        int getcid(){return cid;}
        int geteid(){return eid;}
        bool getbuy(){return buy;}
        int getduration(){return duration;}
        int getprice(){return price;}
        int getshare(){return share;}
        int getts(){return ts;}
};

bool operator<(const order &a,const order &b);
#endif
```

order.h

```
#include "order.h"
bool operator<(const order &a,const order &b){
    if (a.buy){
        if (a.price==b.price) return a.oid>b.oid;
        return a.price<b.price;
    }
```

```
    if (a.price==b.price) return a.oid>b.oid;
        return a.price>b.price;
}
```

order.cpp

```
#ifndef ORDERBOOK_H
#define ORDERBOOK_H
#include <vector>
#include <unordered_map>
#include <string>
#include "equity.h"
#include "client.h"
class orderbook{
    std::unordered_map<std::string,int> ename,cname;
    std::priority_queue<std::string,std::vector<std::string>,std::greater<std::
        string>> ordered_ename,ordered_cname;
    std::vector<equity> equities;
    std::vector<client> clients;
    std::vector<std::string> tttnames;
    bool midpoint,median,transfer,verbose;
    int timestamp,income,transferred,completed,cshare;
    public:
        order order_generate(int oid,std::string line);
        orderbook(bool midpoint,bool median,bool transfer,bool verbose,std::vector<
            std::string> tttnames):
            midpoint(midpoint),median(median),transfer(transfer),verbose(verbose),
                tttnames(tttnames),
            ename(),cname(),ordered_ename(),ordered_cname(),equities(),clients(),
                timestamp(0),income(0),transferred(0),completed(0),cshare(0){}
        void order_execute(order neworder);
        void endofday();
        void endoftime(int ts=-1);
        int getts(){return timestamp;}
};
#endif
```

orderbook.h

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

#define MIN(a,b) a<b?a:b

order orderbook::order_generate(int oid,std::string line){
    std::istringstream sin(line);
    std::string tmp;
    int cid,share,duration,price,eid,ts;
    bool buy;
    sin >> tmp;
    ts = std::stoi(tmp);
    if (ts>timestamp) endoftime(ts);
    sin >> tmp;
    if (cname.find(tmp)==cname.end()){
        ordered_cname.push(tmp);
        cid = cname.size();
```

```
19          cname[tmp] = cid;
20          clients.emplace_back(cid,tmp,0);
21      }
22      else cid = cname[tmp];
23      sin >> tmp;
24      buy = (tmp=="BUY");
25      sin >> tmp;
26      if (ename.find(tmp)==ename.end()){
27          ordered_ename.push(tmp);
28          eid = ename.size();
29          ename[tmp] = eid;
30          equities.emplace_back(eid,tmp);
31      } else eid = ename[tmp];
32      sin >> tmp;
33      price = std::stoi(tmp.substr(1));
34      sin >> tmp;
35      share = std::stoi(tmp.substr(1));
36      sin >> tmp;
37      duration = std::stoi(tmp);
38      if (duration!=-1) duration+=ts;
39      sin.clear();
40      return order(oid,cid,eid,buy,share,duration,ts,price);
41  }
42
43  void orderbook::order_execute(order neworder){
44      equity &eq = this->equities[neworder.geteid()];
45      eq.modify(timestamp,neworder.getprice(),neworder.getbuy());
46      client &c1 = this->clients[neworder.getcid()];
47      if (neworder.getbuy()){
48          while (neworder.getshare()!=0){
49              if (eq.sellercount()==0){
50                  if (neworder.getduration()==-1||neworder.getduration()>this->
                        timestamp)
51                      eq.addbuyer(neworder);
52                  //eq.modify(timestamp);
53                  return;
54              }
55              order seller = eq.getseller();
56              while (seller.getduration()!=-1&&seller.getduration()<=this->timestamp)
                    {
57                  if (eq.sellercount()==0){
58                      if (neworder.getduration()==-1||neworder.getduration()>this->
                            timestamp)
59                          eq.addbuyer(neworder);
60                      //eq.modify(timestamp);
61                      return;
62                  }
63                  seller = eq.getseller();
64              }
65              client &c2 = this->clients[seller.getcid()];
66              if (seller.getprice()<=neworder.getprice()){
67                  int dshare = MIN(seller.getshare(),neworder.getshare());
68                  seller.doneshare(dshare);
69                  neworder.doneshare(dshare);
70                  int commision = seller.getprice()*dshare/100;
71                  this->income+=2*commision;
72                  c2.sell(dshare,seller.getprice()*dshare);
73                  c1.buy(dshare,seller.getprice()*dshare);
```

```
74                  this->transferred+=seller.getprice()*dshare;
75                  this->completed++;
76                  this->cshare+=dshare;
77                  eq.adddealtprice(seller.getprice());
78                  if (this->verbose)
79                      std::cout << c1.getname() << " purchased " << dshare << " shares
                            of " << eq.getname() << " from " << c2.getname() << " for $
                           " << seller.getprice() << "/share" << std::endl;
80              }
81              else{
82                  eq.addseller(seller);
83                  if ((neworder.getduration()==-1||neworder.getduration()>this->
                        timestamp)&&(neworder.getshare()!=0))
84                      eq.addbuyer(neworder);
85                  //eq.modify(timestamp);
86                  return;
87              }
88              if ((seller.getduration()==-1||seller.getduration()>this->timestamp)&&(
                    seller.getshare()!=0))
89                  eq.addseller(seller);
90          }
91          if ((neworder.getduration()==-1||(neworder.getduration()>this->timestamp&&
                neworder.getduration()!=neworder.getts()))&&(neworder.getshare()!=0))
92              eq.addbuyer(neworder);
93      }
94      else {
95          while (neworder.getshare()!=0){
96              if (eq.buyercount()==0){
97                  if (neworder.getduration()==-1||neworder.getduration()>this->
                        timestamp)
98                      eq.addseller(neworder);
99                  //eq.modify(timestamp);
100                 return;
101             }
102             order buyer = eq.getbuyer();
103             while (buyer.getduration()!=-1&&buyer.getduration()<=this->timestamp) {
104                 if (eq.buyercount()==0){
105                     if (neworder.getduration()==-1||neworder.getduration()>this->
                            timestamp)
106                         eq.addseller(neworder);
107                     //eq.modify(timestamp);
108                     return;
109                 }
110                 buyer = eq.getbuyer();
111             }
112             client &c2 = this->clients[buyer.getcid()];
113             if (neworder.getprice()<=buyer.getprice()){
114                 int dshare = MIN(buyer.getshare(),neworder.getshare());
115                 buyer.doneshare(dshare);
116                 neworder.doneshare(dshare);
117                 int commision = buyer.getprice()*dshare/100;
118                 this->income+=2*commision;
119                 c2.buy(dshare,buyer.getprice()*dshare);
120                 c1.sell(dshare,buyer.getprice()*dshare);
121                 this->transferred+=buyer.getprice()*dshare;
122                 this->completed++;
123                 this->cshare+=dshare;
124                 eq.adddealtprice(buyer.getprice());
```

```
125                     if (this->verbose)
126                         std::cout << c2.getname() << " purchased " << dshare << " shares
                                of " << eq.getname() << " from " << c1.getname() << " for $
                            " << buyer.getprice() << "/share" << std::endl;
127                 }
128                 else {
129                     eq.addbuyer(buyer);
130                     if (neworder.getduration()==-1||neworder.getduration()>this->
                        timestamp)
131                         eq.addseller(neworder);
132                     //eq.modify(timestamp);
133                     return;
134                 }
135                 if ((buyer.getduration()==-1||buyer.getduration()>this->timestamp)&&(
                    buyer.getshare()!=0))
136                     eq.addbuyer(buyer);
137             }
138             if ((neworder.getduration()==-1||(neworder.getduration()>this->timestamp&&
                neworder.getduration()!=neworder.getts()))&&(neworder.getshare()!=0))
139                 eq.addseller(neworder);
140         }
141         //eq.modify(timestamp);
142     }
143
144     void orderbook::endofday(){
145         std::cout << "---End of Day---\nCommission Earnings: $" << this->income << "\
                nTotal Amount of Money Transferred: $" << this->transferred << "\nNumber of
                Completed Trades: " << this->completed << "\nNumber of Shares Traded: " <<
                this->cshare << std::endl;
146         if (this->transfer){
147             while (!(this->ordered_cname.empty())){
148                 std::string na = ordered_cname.top();
149                 ordered_cname.pop();
150                 client cl = this->clients[cname[na]];
151                 std::cout << na << " bought " << cl.getbought() << " and sold " << cl.
                        getsold() << " for a net transfer of $" << cl.getincome() << std::
                        endl;
152             }
153         }
154         for (auto t:tttnames){
155             equity eq = equities[ename[t]];
156             eq.strategyout();
157         }
158     }
159
160     void orderbook::endoftime(int ts){
161         //for (auto &t:equities) t.modify(timestamp);
162         if (this->median){
163             std::priority_queue<std::string,std::vector<std::string>,std::greater<std::
                    string>> newname(this->ordered_ename);
164             while (!(newname.empty())){
165                 std::string na = newname.top();
166                 newname.pop();
167                 equity eq = this->equities[ename[na]];
168                 int me = eq.getmedian();
169                 if (me>=0){
170                     std::cout << "Median match price of " << na << " at time " <<
                            timestamp << " is $" << me << std::endl;
```

```
171             }
172         }
173     }
174     if (this->midpoint){
175         std::priority_queue<std::string,std::vector<std::string>,std::greater<std::
            string>> newname(this->ordered_ename);
176         while (!(newname.empty())){
177             std::string na = newname.top();
178             newname.pop();
179             equity eq = this->equities[ename[na]];
180             if (eq.buyercount()==0||eq.sellercount()==0)
181                 std::cout << "Midpoint of " << na << " at time " << timestamp << "
                    is undefined" << std::endl;
182             else{
183                 order bu = eq.getbuyer();
184                 bool quit=false;
185                 while (bu.getduration()!=-1&&bu.getduration()<=this->timestamp){
186                     if (eq.buyercount()==0){
187                         std::cout << "Midpoint of " << na << " at time " <<
                            timestamp << " is undefined" << std::endl;
188                         quit=true;
189                         break;
190                     }
191                     bu = eq.getbuyer();
192                 }
193                 if (quit) continue;
194                 order se = eq.getseller();
195                 while (se.getduration()!=-1&&se.getduration()<=this->timestamp){
196                     if (eq.sellercount()==0){
197                         std::cout << "Midpoint of " << na << " at time " <<
                            timestamp << " is undefined" << std::endl;
198                         quit = true;
199                         break;
200                     }
201                     se = eq.getseller();
202                 }
203                 if (quit) continue;
204                 int mp = (bu.getprice()+se.getprice())/2;
205                 eq.addbuyer(bu);
206                 eq.addseller(se);
207                 std::cout << "Midpoint of " << na << " at time " << timestamp << "
                    is $" << mp << std::endl;
208             }
209         }
210     }
211     if (ts!=-1)
212         timestamp = ts;
213     else timestamp++;
214 }
```

orderbook.cpp