Stock management

The choice of the data structures depends basically on the two analysis functions, `getHighestValueStock` and `getHighestGlobalValueDay`. Both require to analyse transactions in an interval of days. This means that transactions need to be stored by day, to minimize the time for search. Since the day is in fact a key, we can use a hash table (key = day, value = transactions in a day)

The number of transactions per day is unknown, so the transactions in a day should be stored in a dynamic data structure (linked or resizeable array). Since the two get functions need to analyze all transactions in a day, a linked list (simpler than the resizeable array) is a reasonable choice

Pseudocode for data structures

Transaction:

```
struct transaction{
    char* stockID;
    int nShares;
    double value;
    long int day;
} transactionStruct;
```

List of transactions in a day:

```
struct listDayTransactions {
    transactionStruct *data;
    struct listDayTransactions *next;
} listDayTransactions;
```

Hash table day – transactions per day:

```
struct hashtable {
    key_type key;
    value_type value;
    struct hash_el_tag* next;
} hashtableStruct;
```

Pseudocode for functions and complexity analysis

- `void setStockDay(char* stockID, int nShares, double value, long int day){`

```
transactionStruct t = createTransaction(stockID, nShares, value)

listDayTransactions = get(hashTableStruct, day);
```

insertAtHEadOfList(listDayTransaction, t);

time complexity: O(1) (from get) + O(1) (from addToHead)

- `char* getHighestValueStock(long int fromDay, long int toDay);`

we need to access many days, in sequence, in the hash table. This can be done by creating the key for each day *fromDay toDay* . This requires to add 1 to fromDay, and increment the month when needed.

while (day = fromDay until today){

```
listDayTransactions = get(hashTableStruct, day);

scan all transactions in listDayTransactions and find maxValue Stock

find max on all days
```

}

time complexity: O(nTransactionsInADay) (from find max value in a day) *
O(numberOfDays) (from while) (remark that get on hash table is O(1))

- `long int getHighestGlobalValueDay(long int fromDay, long int toDay);`

here the algorithm is in fact a variant of the previous one

while (day = fromDay until today){

```
listDayTransactions = get(hashTableStruct, day);

scan all transactions in listDayTransactions and compute valueDay +=
            nShares * value

find day with max valueDay
```

}

time complexity: O(nTransactionsInADay) (from compute valueDay) *
O(numberOfDays) (from while) (remark that get on hash table is O(1))