Quick note: I’m not too confident about what I have built this time. If you have time off and are willing to go over a tutorial on this through a youtube video (like you did previously) that would be extremely helpful to me! Thank you!

Source Code:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

struct NodeType;

//typedef to define a new data type for the pointer

typedef NodeType \*NodePtr;

//this makes it so that if we declare something with Nodetype

//it automatically becomes a pointer

//structure definition for the record types

struct record

{

long id;

string fname;

string lname;

double amount;

};

//structure definition for the info to be stored in the linked list

struct NodeType

{

long id;

string fname;

string lname;

double amount;

NodePtr flink;

NodePtr blink;

};

//class for the list of user accounts

class AccountList

{

//public functions

public:

AccountList(); //constructor declaration

//function declarations for the tasks to be done

void addAccountSorted (record rec);

void updateAccount (record rec);

void display(ofstream &lfstream);

//private variables

private:

NodePtr head; //pointer to the head of the linkedlist

NodePtr cursor; //just a pointer

};

AccountList::AccountList()

{

//constructs the dummy node

head = new NodeType;

head->id = -1;

head->fname = "";

head->lname = "";

head->amount= -999.999;

head->flink=head; //points the head to itself

head->blink=head; //points the tail to itself

cursor = head; //points the external cursor to the head

}

void AccountList::addAccountSorted(record rec)

{

NodePtr newPtr = new NodeType; //dynamically allocate memory for the new pointer

//inserting all the record details

newPtr->id = rec.id;

newPtr->fname = rec.fname;

newPtr->lname = rec.lname;

newPtr->amount = rec.amount;

newPtr->flink = NULL;

newPtr->blink = NULL;

NodePtr cur, prev;

//loop to iterate through the linkedlist

for (cur=head->flink; cur!=head; cur=cur->flink)

{

if (rec.id < cur->id) //stops the loop when the place is found

break;

}

//cur is now pointing to the Point of Insertion node.

//Using cur, initialize prev that will point to the node just before the cur node.

prev = cur -> blink;

//Create the two forward links

newPtr -> flink = prev -> flink;

prev -> flink = newPtr;

//Create the two back links

newPtr -> blink = cur -> blink;

cur -> blink = newPtr;

//The new node is now inserted just before the Point Of Insertion Node.

}

void AccountList::updateAccount(record rec)

{

if (cursor == head) // if cursor is at dummy node

cursor = cursor->flink; //move to the first node

if (cursor->id == rec.id) //if cursor is at the target node

{

//perform the updates

cursor->fname = rec.fname;

cursor->lname = rec.lname;

if(rec.amount > 0)

{

cursor->amount += rec.amount;

}

else

{

cursor->amount += rec.amount;

}

//remove the account

if(cursor->amount <=0)

{

NodePtr temp = cursor;

cursor->blink->flink = cursor->flink;

cursor->flink->blink = cursor->blink;

cursor = cursor->flink;

delete(temp);

}

}

//if the target node is in the forward direction

else if (cursor->id < rec.id)

{

//locate the target node or point of insertion node

while (cursor != head)

{

if (cursor->id >= rec.id)

break;

cursor = cursor->flink;

}

//if target node is found

if (cursor->id == rec.id)

{

//perform the same update operation

cursor->fname = rec.fname;

cursor->lname = rec.lname;

if(rec.amount > 0)

{

cursor->amount += rec.amount;

}

else

{

cursor->amount += rec.amount;

}

if(cursor->amount <=0)

{

NodePtr temp = cursor;

cursor->blink->flink = cursor->flink;

cursor->flink->blink = cursor->blink;

cursor = cursor->flink;

delete(temp);

}

}

else

{

//initialize the new node

NodePtr newPtr = new NodeType;

newPtr->id = rec.id;

newPtr->fname = rec.fname;

newPtr->lname = rec.lname;

newPtr->amount = rec.amount;

newPtr->flink = NULL;

newPtr->blink = NULL;

//insert the new node before the cursor node

newPtr->blink = cursor->blink;

newPtr->flink = cursor;

cursor->blink->flink = newPtr;

cursor->flink->blink = newPtr;

}

}

//last case when the target node is moving backwards

else

{

while (cursor != head ) //when cursor hasnt reached the dummy node

{

if (cursor->id <= rec.id)

break;

cursor = cursor->blink;

}

//when target node is found

if (cursor->id == rec.id)

{

cursor->fname = rec.fname;

cursor->lname = rec.lname;

if(rec.amount > 0)

{

cursor->amount += rec.amount;

}

else

{

cursor->amount += rec.amount;

}

if(cursor->amount <=0) //delete the account if balance is less than 0

{

NodePtr temp = cursor;

cursor->blink->flink = cursor->flink;

cursor->flink->blink = cursor->blink;

cursor = cursor->flink;

delete(temp);

}

}

else

{

//move the cursor one node

NodePtr newPtr = new NodeType;

newPtr->id = rec.id;

newPtr->fname = rec.fname;

newPtr->lname = rec.lname;

newPtr->amount = rec.amount;

newPtr->flink = NULL;

newPtr->blink = NULL;

//insert the node before the cursor node

cursor = cursor->flink;

newPtr->blink = cursor->blink;

newPtr->flink = cursor;

cursor->blink->flink = newPtr;

cursor->flink->blink = newPtr;

}

}

}

void AccountList::display(ofstream & lfout)

{

//display the results

for(NodePtr cur = head->flink; cur!=head; cur=cur->flink)

lfout << cur->id << " " <<

cur->fname << " " << " " << cur->lname <<

" " << cur->amount << endl;

}

int main()

{

//initialize the records and account list

record recType;

AccountList accounts;

//declare and initialize the output file

ofstream fout ("log.txt");

//declare variables to hold the user input for the name of the file

string m;

string t;

//get user input

cout<<" Enter the master file :";

cin>>m;

cout<<" Enter the transaction file :";

cin>>t;

//open the file according to the user input string

ifstream m\_in(m.c\_str());

ifstream t\_in(t.c\_str());

//if file is opened successfully

if(m\_in.is\_open())

{

//while the master file is not at the end

while(!m\_in.eof())

{

m\_in>>recType.id;

m\_in>>recType.fname;

m\_in>>recType.lname;

m\_in>>recType.amount;

accounts.addAccountSorted(recType);

}

//display the list of values from the linked list by

//calling the function

accounts.display(fout);

if(t\_in.is\_open())

{

while(!t\_in.eof())

{

t\_in>>recType.id;

t\_in>>recType.fname;

t\_in>>recType.lname;

t\_in>>recType.amount;

accounts.updateAccount(recType);

}

accounts.display(fout);

}

else

cout<<"ERROR: unable to open "<<t;

}else

cout<<"ERROR: unable to open "<<m;

m\_in.close();

t\_in.close();

fout.close();

}

Contents of the log

10203 Mindy Ho 2000

12345 Jeff Lee 211.22

14142 James Bond 1500

20103 Ed Sullivan 3000

22345 Norma Patel 2496.24

27183 Teresa Wong 1234.56

30102 Ray Baldwin 3824.36

30201 Susan Woo 9646.75

31456 Jack Smith 1200

31623 Norris Hunt 1500

10101 Judy Malik 800

12345 Jeff Lee 211.22

14142 James Bond 3000

20103 Ed Sullivan 3000

20301 Joe Hammil 500

22222 Joanne Doe 2750.02

30102 Ray Baldwin 3824.36

30201 Susan Woo 9646.75

31416 Becky Wu 200

32123 John Doe 900

Contents of the master

27183 Teresa Wong 1234.56

12345 Jeff Lee 211.22

31456 Jack Smith 1200.00

14142 James Bond 1500.00

31623 Norris Hunt 1500.00

10203 Mindy Ho 2000.00

20103 Ed Sullivan 3000.00

30102 Ray Baldwin 3824.36

30201 Susan Woo 9646.75

22345 Norma Patel 2496.24