Bank account manager stimulator

**I. Problem:**

Design a data structure to stimulate managing bank accounts. Each account consists of a unique username, the amount of money currently in the account, and a list of transactions performed in history. Each transaction consists of an id which is different for every transaction, the action performed (withdraw, deposit, ...), and the amount of money transferred.

Both accounts and transactions have a unique value, which are username and id, so the tree structures can be effective.

The number of accounts can be large and the list get updated less often than the transaction, so I chose B-tree for the job. The number of transactions is usually smaller, so I decided to use a binary tree instead. The transaction history is more likely to be updated than to be looked up, which is the reason I chose the red-black tree over AVL to store the history, as red-black tree provides faster insertion and removal operation than AVL (Source: <https://www.geeksforgeeks.org/red-black-tree-vs-avl-tree/> ).

In short, I want to use the b-tree to manage accounts, each node in this b-tree also hold a root node of a red-black tree used to manage transaction history.

**II. Implementation:**

**1. Structures:**

My program defined six structures made out of each other.

Start with a structure to store a transaction. Currently, it can store the transaction id, the action performed, and the amount of money transferred.

struct trans

{

string id;

string action;

int amount;

};

Using the transaction structure as data, the red-black tree node for transaction history is defined.

typedef struct historyNode

{

trans data;

bool color;

historyNode \*left;

historyNode \*right;

historyNode\* parent;

} H;

The historyList structure holds the root node of a red-black tree and the number of nodes in that tree, which is the number of transactions performed history of an account.

struct historyList

{

H\* root;

int count;

};

The structure to store a profile consists of a username, the current amount of money in the account, and a historyList.

struct account

{

string username;

int amount;

historyList \*history;

};

Using the account structure as data, the b-tree node for account manage tree is defined.

typedef struct accountnode

{

account data[WAY - 1];

accountnode\* child[WAY];

int count;

accountnode\* parent;

bool isLeaf;

} A;

Finally, the accountList structure, similar to the historyList, stores the root node of the account b-tree and the number of accounts in that tree.

struct accountList

{

A\* root;

int count;

};

**2.Stored file:**

First I design the form of the file use to store all information. This form is apply for both the input file that the program will read at the beginning, and the output file of the save function.

- First line in the file is a single integer number M , the number of account currently stored in the file.

- The number of transactions of the ith account is Ni .

- Information of each account is written on 1 + Ni lines, start from the second line of the file.

- The first line of each account will have form:

<username> <current money in account> <number of transaction in history (Ni)>

- The next Ni line will have form:

<id> <action> <amount of money>

Example for a single account stored form:



Example for the stored file:



The order of accounts and transactions in the input file may be random as long as they are in the right form. But in the output file of the save file function, the order of accounts is sorted in ascending by their username, and the transactions is sorted in ascending order by their id.

**3.About source code:**

The attached source code provide all operation functions on the b-tree and red-black tree (search, insert, delete).

The main program currently allow these action:

- Read file (done automatically).

- Search for profile with a given username.

- Search for specific transaction in profile with a given username and id.

- Create new empty profile (only have a username).

- Remove an existing profile with a given username.

- Save all information to a file.

More action functions is still being work on.