CS302

Assignment 6: It's in the bank

Description

Banks. We had a bank crisis over a decade ago and recently another one in Silicone Valley. However, for our sort of bank database we will implement, everything will go smoothly. We will use a fast search structure to look up bank account records, using a custom `hashMap` class, the details are shown below:

```
template <class t1, class t2>
class hashMap
{
public:
  hashMap(std::size_t = 5);
  t2& operator[](t1);
private:
  struct keyVal
  {
    keyVal()
    {
      key = t1();
      value = t2();
    }

    t1 key;
    t2 value;
  };

  void resize(std::size_t);

  std::size_t hash1(std::string);
  std::size_t hash2(std::string);

  std::vector<keyVal> table1;
  std::vector<keyVal> table2;

  std::size_t items1;
  std::size_t items2;
};
```

Where each member contains/performs the following

- `struct keyVal` - a struct object that contains a (key, value) pair for each entry
- `std::vector<keyVal> table1` - a hash table that stores a set of entries
- `std::vector<keyVal> table2` - a hash table that stores a set of entries
- `std::size_t items1` - a counter that indicates the amount of entries currently stored in table1
- `std::size_t items2` - a counter that indicates the amount of entries currently stored in table2
- `std::size_t hashMap<t1, t2>::hash1(std::string key)` - returns a value for a given key, the function returns the following summation (k denotes the length of the string `key`)

$$\sum_{i=0}^{k}(key_i \times 10^i)$$

- `std::size_t hashMap<t1, t2>::hash1(std::string key)` - returns a value for a given key, the function returns the following summation

$$\sum_{i=0}^{k}(key_{(k-1-i)} \times 10^i)$$

- `void hashMap<t1, t2>::resize(std::size_t amount)` - a function that resizes table1 and table2 by amount, all of the elements that were originally in table1 and table2 needs to be rehashed into larger tables (you can use the `operator[]` in this function to help rehash everything)
- `hashMap<t1, t2>::hashMap(std::size_t size)` - constructor that sets `items1` and `items2` with 0, and sizes the two tables with the parameter value (you might need to implement a default constructor for `keyVal`, you can set `key = t1()` and `value = t2()` in the `keyVal` constructor)
- `t2& hashMap<t1, t2>::operator[](t1 key)` - bracket operator function that implements find and insert, the steps are described as the following
  1. Determine the load factor of both tables, if one or both load factors are 20% or higher, resize the tables by doubling their sizes (so you would call `this->resize(table1.size())` we are sizing both tables by table size as the amount so both tables will be doubled in size after resize is done)
  2. `std::size_t index1 = hash1(key) % table1.size()`
     and `std::size_t index2 = hash2(key) % table2.size()`
  3. Initialize two collision counters `i1` and `i2` with 0
  4. Check `table1[index1]`, if the element is blank then create a new entry in this position, increment `items1` by 1 and return `table1[index1].value`, if this entry has a matching key then just return `table1[index1].value`
  5. Perform the same steps as the above step for `table2[index2]`
  6. If a collision occurred (in both tables), then increment both collision counters and then update index1 and index2 in the following way

```
index1 = (index1 + i1 * hash2(key)) % table1.size(); //on even iterations
index1 = (index1 + i1 * hash1(key)) % table1.size(); //on odd iterations

index2 = (index2 + i2 * hash1(key)) % table2.size(); //on even iterations
index2 = (index2 + i2 * hash2(key)) % table2.size(); //on odd iterations
```

  And then go to back step to 4

Contents of Main

Your program is going to maintain a set of bank records and your program needs to lookup records to allow a user to display the record and/or update the balances etc. you can create additional variables and maps as needed but you **MUST** declare and use the following

```
hashMap< std::string, hashMap< std::string, bankType> > bank;
```

Where bankType is a global struct defined in the following way

```
struct bankType
{
  //each bank record will have a litf of transactions
  struct transactionType
  {
    //transaction type true/false deposit/withdrawal
    bool type;
    //transaction amount
    double amount;

    transactionType(bool t, double a)
    {
      type = t;
      amount = a;
    }
  };

  std::string name; //name on account
  std::string acctNo; //bank account number
  double balance; //current balance on acount
  unsigned int pin; //pin number
  bool locked; //true/false for locked/unlocked

  //a list of deposit/withdrawal and amounts of each
  std::vector<transactionType> transactions;


  //Some constructors to help make this struct easier to handle
  bankType(std::string first, std::string last, std::string acctNo,
    double balance,unsigned int pin)
  {
    name = last + ", " + first;
    this->balance = balance;
    this->pin = pin;
    this->acctNo = acctNo;
    locked = false;
  }

  bankType()
  {
    name = "";
    balance = 0;
    pin = 0;
    locked = false;
  }
};
```

The `bank` object will be a 2 way hash structure, so every bank has a location you search for then the last 4 digits of the bank account, so if there if there is a bank account number 1234567890 located at Fort Apache, then you look up the `bankType` object using the following code

```
bank["Fort Apache"]["7890"]
```

And then you can use the dot operator to access that bank's records (the name, account number, pin, etc)

Input

You will be given a CSV file of bank records (you don't need to read in the filename, you can open an input filestream and hardcode the filename, the file is called `"data.csv"`) and you will need to populate the `bank` map, each line of the CSV file is terminated by a linux style end of line character, and each line contains the following

```
"FIRST_NAME,LAST_NAME,LOCATION,ACCT_NO,BALANCE,PIN"
```

Each attribute of a line is separated by a comma, you can use the substr/find/erase string functions to separate the parse the line into comma separated tokens (since there's no split function for strings in C++ and stringstreams is some weird library). For example if the first line of data in the file contains

```
"Idette,Delooze,Spring Mountain,8603817497,18734.63,1411"
```

And we used `std::getline(infile, line)` to read the line into a `string` variable called `line`, then we can do the following to get the first name

```
std::string firstName = line.substr( 0, line.find(",") );
```

So `firstName` contains `"Idette"`, then we can remove this first name from `line` using the following

```
line.erase( 0, line.find(",") + 1 );
```

And then if we apply the following code again

```
std::string lastName = line.substr( 0, line.find(",") );
```

the variable `lastName` contains `"Delooze"`, and you can continue this process until every token (substring between the commas) has been extracted, and then you store this record into `bank` object using the following

```
bank[location][acct4Digits] = bankType(first, last, acctNo, balance, pin);
```

Where `acct4Digits` is a string that contains the last 4 digits (as a string) of `acctNo`

Main Algorithm

Once everything is read from the CSV file, using the `bank` object and any other variables (or maybe additional `hashMap` objects), once all the data structures are set, perform the following steps

1. Ask a location from the user for a location by outputting `"Please enter a bank branch location: "`
2. If the location does not exist, output `"Bank branch not found"` and go to step 14 otherwise continue to the next step
3. Ask for the last 4 digits of the account at the location acquired from the previous step, if the account is not found, output `"Account not found"` then go to step 14 otherwise continue to the next step
4. If the account is locked, output `"Account has a lock"` and go to step 14, otherwise continue to the next step
5. Ask the user for a pin by prompting `"Enter a pin: "`

6. If the pin read from the above step matches `bank[location][acct].pin` then go to step 8
7. The pin is not correct, so output `"Invalid pin"` and update a counter, if the counter has not exceed 3 then go back to step 5, if the counter is maxed out to 3, then lock the account, output `"3 failed login attempts, account is now locked"` and go to step 14
8. Prompt the user an option and go to the next step
9. If the user enters `'D'` or `'d'`, prompt the user for a deposit amount, add this amount the the account's balance and insert `{true, depositAmount}` into the user's transactions list and go to step 8
10. If te user enters `'w'` or `'W'`, prompt the user for a withdrawal amount, subtract this amount from the account's (balance is less than withdrawal amount, output `"Not enough funds to make transaction"`) and go to step 8, otherwise subtract the amount from the balance and insert `{false, withdrawalAmount}` into this accounts transactions list and then go to step 8
11. If the user enters `'V'` or `'v'`, output the account name, the balance and all of the transactions (if no transactions have been performed on this account then output `"No transactions"`), then go to step 8
12. If the user enters `'E'` or `'e'` then go to step 14
13. If the enters an invalid option, output `"Invalid choice"` and then go to step 8
14. Prompt the user if they wish to continue, if `'Y'` or `'y'` then go to 1, if the user enters `'N'` or `'n'` then end the program

Specifications

- Add header comments in your header file and main file (and add block comments for the larger functions, and have maybe inline comments for the smaller functions)
- Do not use linear search, all searches need to use the `hashMap` bracket operator function
- Do not set the initial size of the `hashMap` to a different size other than the default parameter of 5 (we want to test out that your resize works), but for debugging purposes of course you can create an arbitrarily large table size just to make sure your `hashMap` search/insert is working
- Do not use `std::unordered_map` in your main, of course you can use `std::unordered_map` to make sure your main algorithm works but if you submit using this library, a major deduction of points will be applied to your assignment

Example Output

```
% g++ main.cpp
% ./a.out

Please enter bank branch location: Rainbow

Enter last 4 digits of account: 6759

Enter a pin: 1111

Invalid pin

Enter a pin: 2546

Invalid pin
```

```
Enter a pin: 8459

(D)eposit
(W)ithdrawal
(V)iew account
(E)xit account
Enter choice: v

Name: Broy, Baudoin
Balance: $2724.62

No transactions

(D)eposit
(W)ithdrawal
(V)iew account
(E)xit account
Enter choice: W

Enter withdrawal amount: 200.56

(D)eposit
(W)ithdrawal
(V)iew account
(E)xit account
Enter choice: D

Enter deposit amount: 150.56

(D)eposit
(W)ithdrawal
(V)iew account
(E)xit account
Enter choice: E

Continue? (Y/N): y

Please enter bank branch location: rainbow

Enter last 4 digits of account: 5555
Account not found

Continue? (Y/N): y

Please enter bank branch location: rainbow

Enter last 4 digits of account: 6759

Enter a pin: 8459

(D)eposit
(W)ithdrawal
(V)iew account
```

```
(E)xit account
Enter choice: V

Name: Broy, Baudoin
Balance: $2674.62

List of transactions
Withdrawal amount $200.56
Deposit amount $150.56

(D)eposit
(W)ithdrawal
(V)iew account
(E)xit account
Enter choice: W

Enter withdrawal amount: 10000.00

Not enough funds to make transaction

(D)eposit
(W)ithdrawal
(V)iew account
(E)xit account
Enter choice: e

Continue? (Y/N): y

Please enter bank branch location: rainbow

Enter last 4 digits of account: 6759

Enter a pin: 1111

Invalid pin

Enter a pin: 2222

Invalid pin

Enter a pin: 3333

Invalid pin

3 failed login attempts, account is now locked

Continue? (Y/N): y

Please enter bank branch location: rainbow

Enter last 4 digits of account: 6759
Account has a lock

Continue? (Y/N): y
```

```
Please enter bank branch location: Hendertucky

Bank branch not found

Continue? (Y/N): n
```

Submission

Submit the source code to code grade by the deadline

References

- Link to the top image can be found at $https://southpark.cc.com/news/zi5uql/aannnd-it-s-gone$
- Supplemental Video https://youtu.be/R7bydnR6zCw