CSCI3060U Project Assignment #1

Front End Requirements

Taylor Smith
Alexandar Mihaylov
Talha Zia

Question1. A list of all your test cases and what they are intended to test (a table of test names and intentions, in English)

List of Test Cases and What They are to Test

Descriptions - **2 marks**

- table of test cases telling which case each is intended to test

Completeness - **2 marks**

- test cases cover majority of cases allowed by requirements, and include cases to test all constraints

Login

login01	Check all commands **do not work** before user logs in: logout, withdrawal, transfer, paybill, deposit, create, delete, disable, changeplan, and check for gibberish
login02	Check that you can login as a **standard** user, logout successfully and an “empty” transaction file is generated.
login03	Check that you can login as an **admin** user, logout successfully and an “empty” transaction file is generated.
login04	Check that after logging in as a **standard** user you **cannot** login again.
login05	Check that after logging in as an **admin** user you **cannot** login again.
login06	Check that a **standard** user can login and then use **all** unprivileged commands.
login07	Check that a standard user can login and **cannot** use **all** privileged commands.
login08	Check that an admin user **can** use **all** privileged commands

login09	Bad input before login, and after login fails gracefully
login10	User enters an account holder name that does not exist in the current accounts file, user is given an error

Logout

logout01	Check that a user cannot logout before they have logged in
logout02	Once a user has logged in and performed **0 transactions**, then logs out, check that there was nothing regarding transactions written to the transaction file
logout03	Once a user has logged in and performed **1 transaction**, then logs out, assert that transaction was written to the transaction file
logout04	Once a user has logged in and performed **2 or more transactions**, then logs out, assert that transactions were written to the transaction file
logout05	User performs a login and logout session, then is able to log in once again followed by a successfull logout
logout06	After logging in, use is unable to **login** before they logout of their current session

Withdrawal Test

withdrawal01	The account holder's name and account number match and withdrawal proceeds as intended
withdrawal02	The account holder's name and account number **do not** match, error thrown and user is sent back to main menu

withdrawal03	User supplies a **negative** value to withdrawal and is rejected
withdrawal04	The user supplies random input (ex: #badinput!) into the amount to withdraw that is not in the form **\d+\.\d{2}** and is **rejected**
withdrawal05	User supplies 1.00 value to withdrawal and is rejected since it is not a paper currency value
withdrawal06	User supplies 5.00 value to withdrawal and is accepted since it is a multiple of 5
withdrawal07	A **standard user** supplies 400.00 dollars to withdrawal and is accepted since it is under the 500 limit
withdrawal08	A **standard user** supplies 500.00 dollars to withdrawal and is accepted since it is under the 500 limit (account has more than $500.10 in it)
withdrawal09	A **standard user** supplies 505.00 dollars to withdrawal and is **rejected** since it is above the 500 limit
withdrawal10	An **admin user** supplies 505.00 dollars to withdrawal and is **accepted** since admins do not have limits
withdrawal11	A **standard user** supplies 400.00 dollars to withdrawal then after attempting to withdrawal another 105.00 in the same day, is **rejected** and given an error since the combined amount is > 500.00
withdrawal12	An **admin** supplies 400.00 dollars to withdrawal then after attempting to withdrawal another 105.00 in the same day, is **accepted**

withdrawal13	A **standard user** supplies 400.00 dollars to withdrawal then after attempting to withdrawal another 100.00 in the same day, is **accepted** (account has more >= 500.20 in it)
withdrawal14	Standard student account withdrawals 5.00 from an account and is **accepted**
withdrawal15	Standard student account withdrawals 5.00 from an account with 5.05 in it and is **accepted**
withdrawal16	Standard student account withdrawals 5.00 from an account with 5.00 in it and is **rejected**
withdrawal17	admin student account withdrawals 5.00 from an account with 5.00 in it and is **accepted**
withdrawal18	Standard non-student account withdrawals 5.00 from an account and is **accepted**
withdrawal19	Standard non-student account withdrawals 5.00 from an account with 5.10 in it and is **accepted**
withdrawal20	Standard non-student account withdrawals 5.00 from an account with 5.05 in it and is **rejected**
withdrawal21	admin non-student account withdrawals 5.00 from an account with 5.05 in it and is **accepted**

Transfer

transfer01	The first account (sender) number and account nameholder **match**
transfer02	The first account (sender) number and account nameholder **do not** match and an error is thrown

transfer03	The second account (receiver) supplied by the user exists and user proceeds as normal
transfer04	The second account (receiver) supplied by the user **does not** exist
transfer05	User supplies a **negative** value to transfer and is **rejected**
transfer06	The user supplies random input (ex: #badinput!) into the amount to transfer that is not in the form **\d{5}\.\d{2}** and is **rejected**
transfer07	A **standard user** supplies 1.00 dollars to transfer and is accepted since it is under the 1000.00 limit
transfer08	A **standard user** supplies 1000.00 dollars to transfer and is accepted since it is under the 1000.00 limit (account has more than 1000.10 in it)
transfer09	A **standard user** supplies 1001.00 dollars to transfer and is **rejected** since it is above the 1000.00 limit
transfer10	An **admin user** supplies 1001.00 dollars to transfer and is **accepted** since admins do not have limits
transfer11	A **standard user** supplies 900.00 dollars to transfer then after attempting to transfer another 101.00 in the same day, is **rejected** and given an error since the combined amount is > 1000.00
transfer12	An **admin** supplies 900.00 dollars to transfer then after attempting to transfer another 101.00 in the same day, is **accepted**

transfer13	A **standard user** supplies 900.00 dollars to transfer then after attempting to transfer another 100.00 in the same day, is **accepted** (account has more >= 1000.10 in it)
transfer14	Standard student account transfers 5.00 from an account and is **accepted**
transfer15	Standard student account transfers 5.00 from an account with 5.05 in it and is **accepted**
transfer16	Standard student account transfers 5.00 from an account with 5.00 in it and is **rejected**
transfer17	admin transfers 5.00 from a student account with 5.00 in it and is **accepted**
transfer18	Standard non-student account transfers 5.00 from an account and is **accepted**
transfer19	Standard non-student account transfers 5.00 from an account with 5.10 in it and is **accepted**
transfer20	Standard non-student account transfers 5.00 from an account with 5.00 in it and is **rejected**
transfer21	admin account transfers 5.00 from a non-student account with 5.00 in it and is **accepted**
transfer22	Transfers must be strictly greater than 0. Should be **rejected**

Pay Bill

paybill01	Account name and number **match** from the current accounts file

paybill02	Account name and number **do not match** from the current accounts file, user is sent back to the menu
paybill03	User provides “EC” as the company name once prompted and is **able** to proceed with paybill
paybill04	User provides “CQ” as the company name once prompted and is **able** to proceed with paybill
paybill05	User provides “TV” as the company name once prompted and is **able** to proceed with paybill
paybill06	User provides “AA” as the company name once prompted and is **unable** to proceed with paybill since “AA” is not in the list of companies, user is shown an error and taken back to the main menu
paybill07	User supplies a **negative** value to pay and is **rejected**
paybill08	The user supplies random input (ex: #badinput!) into the amount to pay that is not in the form **\d+\.\d{2}** and is **rejected**
paybill09	A **standard user** supplies 1.00 dollar to pay and is accepted since it is under the 2000.00 limit
paybill10	A **standard user** supplies 2000.00 dollars to pay and is accepted since it is under the 2000.00 limit (account has more than $2000.10 in it)
paybill11	A **standard user** supplies 2001.00 dollars to pay and is **rejected** since it is above the 2000.00 limit

paybill12	An **admin user** supplies 2001.00 dollars to pay and is **accepted** since admins do not have limits
paybill13	A **standard user** supplies 1900.00 dollars to pay then after attempting to pay another 101.00 to the **same** bill holder in the same day, is **rejected** and given an error since the combined amount is > 2000.00
paybill14	A **standard user** supplies 1900.00 dollars to pay then after attempting to pay another 101.00 to the **another** bill holder in the same day, is **accepted**
paybill15	An **admin** supplies 1900.00 dollars to pay then after attempting to pay another 101.00 to the same bill holder in the same day, is **accepted**
paybill16	A **standard user** supplies 1900.00 dollars to pay then after attempting to pay another 100.00 in the same day, is **accepted** (account has more >= 2000.10 in it)
paybill17	Standard student account pays 5.00 from an account with sufficient funds is **accepted**
paybill18	Standard student account pays 5.00 from an account with 5.05 in it and is **accepted**
paybill19	Standard student account pays 5.00 from an account with 5.00 in it and is **rejected**
paybill20	admin account pays 5.00 from a student account with 5.00 in it and is **accepted**
paybill21	Standard non-student account pays 5.00 from an account with sufficient funds and is **accepted**

paybill22	Standard non-student account pays 5.00 from an account with 5.10 in it and is **accepted**
paybill23	Standard non-student account pays 5.00 from an account with 5.00 in it and is **rejected**
paybill24	admin non-student account pays 5.00 from an account with 5.00 in it and is **accepted**

Deposit

deposit01	Account name and number **match** from the current accounts file
deposit02	Account name and number **do not match** from the current accounts file, user is sent back to the menu
deposit03	Standard account deposits 0.01 into an account with $0.00 balance, and is **rejected** since it does not cover the 0.1/0.05 fee, user is taken back to menu
deposit04	admin account deposits 0.01 into an account with $0.00 balance, and is **accepted**
deposit05	User deposits 5.20 into their account of 0.00 balance and within the same session attempts to withdrawal 5.00 but is **rejected**
deposit06	A deposit of 1.00 is added to an account with a balance of 99999.99 it should be **rejected** since it exceeds the maximum value an account can hold.

Create

create01	Once prompted to enter account holder name, enter in a string of characters “A” that are of length 20. Check that the

	system created the account holder with 20 “A”s
create02	Once prompted to enter account holder name, enter in a string of characters “A” that are of length 21. Check that the system created the account holder and truncated it to only 20 “A”s
create03	When prompted to enter in an initial balance, input 99999.99 and the transaction should continue without any error
create04	When prompted to enter in an initial balance, input 100000. and the transaction should **reject** it, display an error and send the user back to main menu
create05	Create an account from start to finish, and within the same session attempt to withdrawal, transfer from/to, paybill, and deposit to the created account

Delete

delete01	Once the user is prompted to enter the account name to be deleted, fill it with one that exists in the current bank accounts file and the transaction should **proceed** The user enters in a bank account number that **matches** the account holder name based on the current bank accounts file The account is deleted.
delete02	Once the user is prompted to enter the account name to be deleted, fill it with one that **does not** exist in the current bank accounts file and the transaction should be **rejected**
delete03	The user enters in a bank account number that **does not match** the account holder

	name based on the current bank accounts file
delete04	Once an account deletion has been deleted, and within the same session attempt to withdrawal, transfer from/to, paybill, and deposit to the created account, this should result in a **rejection** and an error by the system

Disable

disable01	Once prompted to enter accountholder name, provide one that **exists** and the user should be able to proceed with the transaction. Provide an account number that matches the accountholder name provided, transaction should proceed as intended
disable02	Once prompted to enter accountholder name, provide one that **does not** exist and the transaction should be **rejected** and the user taken to the main menu
disable03	Provide an account number that **does not** matche the accountholder name provided, transaction should be **rejected** and user taken to the main menu
disable04	After an account has been disabled, within the same session attempt to withdrawal, transfer, paybill, deposit, from that account. All transactions should be **rejected** gracefully with their respective errors.
disable05	An account that exists and is already disabled is attempted to be disabled again.

Enable

enable01	Once prompted to enter accountholder name, provide one that **exists** and the user should be able to proceed with the transaction. Provide an account number that matches the accountholder name provided, transaction should proceed as intended
enable02	Once prompted to enter accountholder name, provide one that **does not** exist and the transaction should be **rejected** and the user taken to the main menu
enable03	Provide an account number that **does not** match the accountholder name provided, transaction should be **rejected** and user taken to the main menu
enable04	After an account has been enabled, within the same session attempt to withdrawal, transfer, paybill, deposit, from that account. All transactions should work as intended as if the account was never disabled.
enable05	An account that exists and is already enabled is attempted to be enabled again.

Change Plan

changeplan01	Once prompted to enter accountholder name, provide one that **exists** and the user should be able to proceed with the transaction. Provide an account number that matches the accountholder name provided, transaction should proceed as intended
changeplan02	Once prompted to enter accountholder name, provide one that **does not** exist and the transaction should be **rejected** and the user taken to the main menu
changeplan03	Provide an account number that **does not** match the accountholder name provided,

	transaction should be **rejected** and user taken to the main menu
changeplan04	After a successful changeplan of an account, within the same session run the changeplan transaction again on the same account

Question 2

For each test case, the actual test input files and expected output files (as text file printouts)

Actual Test Case Inputs and Expected Outputs

Test Files - **2 marks**

- all provided, and with expected output?

***Note: We made the assumption that each test being run is run in its own sandbox environment, which means that if one test withdrawals $5.00 from one account, then the next test is not affected by the previous test. Each test assumes that the *Current Bank Accounts File* is as given at the beginning at every test.**

See included files in the **/tests** directory.
Current Bank Account File: **test.cbaf**

```
/tests
├── 00-login
│   ├── in
│   ├── out
│   └── trans
├── 01-logout
│   ├── in
```

```
│   ├── out
│   └── trans
├── 02-withdrawl
│   ├── in
│   ├── out
│   └── trans
├── 03-transfer
│   ├── in
│   ├── out
│   └── trans
├── 04-paybill
│   ├── in
│   ├── out
│   └── trans
├── 05-deposit
│   ├── in
│   ├── out
│   └── trans
├── 06-create
│   ├── in
│   ├── out
│   └── trans
├── 07-delete
│   ├── in
│   ├── out
│   └── trans
├── 08-disable
│   ├── in
│   ├── out
│   └── trans
├── 09-enable
│   ├── in
│   ├── out
│   └── trans
└── 10-changeplan
    ├── in
    ├── out
    └── trans
```

Question 3

A test plan document, outlining how your tests are organized (in directories or whatever), how they will be run (as shell scripts or DOS batch files), and how the output will be stored and organized for reporting and comparison with later runs (make text file printouts of any directory structures and script files created)

Test Plan

Test Input and Output Organization - **2 marks**

- there is a good plan to keep test cases and results well organized

Test Run Plan - **1 mark**

- there is a workable plan for running all test cases

The test cases are broken down by the command they test. They all use the same current accounts file which is stored in the top level of the directory(test.cbaf). They are organized in a directory structure like so:

```
COMMAND
|-- in
|    |-- INPUT_FILES_FOR_EACH_TEST
|-- out
|    |-- OUTPUT_FILES_FOR_EACH_TEST
|-- trans
|   |-- TRANSFER_FILES_FOR_EACH_TEST
```

Each command directory is labeled in the order it is to be tested(**not** the transaction code for that command) and the files are labeled in the order to be tested as well.

When the tests are run they will be run using a shell script of similar structure to this:

```
for dir in directories //commands
        for file in dir/input
                ./Front_End.exe << file >> {DATE}/out/test_name.out

                if(test_passed)
                        do passing things
                else
                        do failing things
```

The output files are stored in the same structure as the test cases except with the parent directory being the current date(incremented if tests are run multiple times per day, ex. 16/02/14-01). The output results and transaction files can easily be compared and the results are to be displayed to the terminal as well as maintained in a text file in the top level of the directory structure named results.txt

Problems in requirements

Other

Requirements Problems - **1 mark**

- problems found in the requirements are reported

Most problems have been submitted to the client and clarified, here are a few of the remaining ones that have yet to be clarified and the assumptions made when creating test cases:

What happens to available funds when account is deleted?
Assumption: Nothing.

In the transaction file is the misc information flag used to hold the company when the paybill transaction is completed.
Assumption: Yes.

What happens if you try to deposit an amount that puts your account over the 99999.99 limit.
Assumption: It fails.

What happens when you try to withdrawal, paybill, or deposit $0.00.
Assumption: It is rejected