TEST PLAN

TEST INPUT AND OUTPUT ORGANIZATION

We have created a directory named `test_case_files` that contains all the tests' input and output files; these files are then be grouped and divided into the different commands to be tested (`login`, `withdrawal`, `transfer`, `paybill`, `deposit`, `create`, `delete`, `disable`, `changeplan`, and `logout`) – except for the `current_bank_accounts` file, which is also part of this directory.
Two additional commands were created: `help` and `quit`. `help` will aid the users remember which commands are available - as well as their details -, while `quit` will enable the user to exit the system.

As mentioned above, each of this commands will be given a separate directory. Inside each of these directories, there will be four more: `input`, `expected_output`, `real_output`, `transaction` and `real_transaction`. The `input`, `expected_output` and `transaction` directories contain pre-prepared testing files, whereas the `real_output` directory will contain the actual results of the test when the program is ran. These are some examples of what the general structure looks like:

```
luisarojas [test_case_files] $ tree .
.
├── current_bank_accounts_file.txt
├── login
    ├── expected_output
    ├── expected_transaction
    ├── input
    ├── real_output
    └── real_transaction
```

With regards to naming conventions, we decided to use `.in`, `.out` and `.tra` for input, output and transaction files, respectively. Each was named using the command they are meant to be testing, followed by a number – which corresponds to the step being tested on the test cases – and then a letter, which represents one of the variations for that specific test (e.g. `login05b.in`).

TEST RUN PLAN

The tests created will be ran on our Banking System using a shell script, who's output will be stored in the `real_output` directory, as mentioned above, and compared with the `expected_output` files using the `diff` command.

When using the `diff` command, if a non-empty file is created, then this means the test was failed. Otherwise, the test is considered to have succeeded.

Additionally, it was decided that every test would be ran in an independent way; meaning that at the whole system terminates after every test using the `quit` command. *Note that the quit command may not be used if there is a user logged into a session.* This allows for certain flexibility regarding the Current Bank Accounts File, since the program won't need to take into account transactions done in previous test – it starts from zero every time.

Finally, there are some commands that must be tested before others, since they may be necessary for other tests which they depend on the working conditions of the tests before them. For this reason, we have agreed to run the tests in the following order:

1. `quit`: Program needs a way to terminate gracefully – will be used in every test.
2. `help`: Available at any point during the execution of the system.
3. `login and logout`: Used in most test cases.
4. `create`: Need to test for accounts recently created.
5. `delete`
6. `disable`: Need to test for immediate action on disabled accounts.
7. `enable`: Need to test for immediate action on enabled accounts.
8. `changeplan`: Must be taken into account for monetary transactions.
9. `Deposit`: Need to test if recently deposited funds may be used.
10. `withdrawal`
11. `transfer`
12. `paybill`