(CSCI 3060/SOFE 3980U - Winter 2016
Course Project Assignment #1 - Front End Requirements

Denesh Parthipan (100536986), Luisa Rojas-Garcia (100518772), Truyen Truong (100516976)
Note: The test environment for all test cases is Linux Operating System.

Test Case Name: Login

Description: Starts a Front end session. Should ask for the kind of session.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT
1 (1)	Check if command is `login`	login **admin** logout	login **standard** **acc_holder** logout	If command is `login`, continue. If it is not an existing command, show error message. User may use the `help` command.
2 (2)	Check if there is an account already in session.			If there is a user already logged in, display an error message. Otherwise, prompt user to enter the account mode.
3 (3)	Check if account mode (`admin` or `standard`) is valid.	login **admin** logout	login **standard** **acc_holder** logout	If `admin` is the input, login is successful. Inform the user they are now logged in and start reading in the current bank accounts file and as well as start accepting transactions. If `standard` is the input, prompt for `acc_holder`. If the input is neither `admin` nor `standard`, show an error message. User is taken back to the previous step and may attempt to login again.
4 (4)	Check if the `acc_holder` exists in the Bank System in the case of a standard user only. `acc_holder` may not be "END_OF_FILE".	*Doesn't apply.*	login **standard** **acc_holder** logout	If `acc_holder` exists and is valid, login is successful. Inform the user they are now logged in and start reading in the current bank accounts file and as well as start accepting transactions. Otherwise, user is taken back to the previous step and may attempt to login again.

Test Case Name: Logout

Description: Ends a Front End session.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT

1
(5)
	Check if command is `logout`	`login`
		`admin`
		`logout`
			If command is `logout`, continue.
			
			If it is not an existing command, show error message. User may use the `help` command.

2 (6)	Check if there is an account in session.		If there is a user already logged in, successfully logout. Otherwise, display an error message.

Test Case Name: Create

Description: Create a new bank account with an initial balance. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT

1
(7)
	Check if the command is `create`.	`login`
		`admin`
		`create`
		`new_acc_holder`
		`init_balance`
		`logout`
			If command is `create`, continue.
			
			If it is not an existing command, show error message. User may use the `help` command.

2
(8)
	Check that user is logged in.	`login`
		`admin`
		`create`
		`new_acc_holder`
		`init_balance`
		`logout`
			If user is logged in, continue.
			
			Otherwise, display an error message.

3
(9)
	Check that current logged in user is an administrator.	`login`
		`admin`
		`create`
		`new_acc_holder`
		`init_balance`
		`logout`
			If account mode is `admin`, prompt user for `new_acc_holder`.
			
			Otherwise, display an error message.

4
(10)
	Check that `new_acc_holder` is a valid name:
	- No longer than 20 characters.
	- Only letters,
		`login`
		`admin`
		`create`
		`new_acc_holder`
		`init_balance`
			If `new_acc_holder` is valid, then continue.
			
			If `new_acc_holder` is longer than 20 characters, truncate account name and then continue.

	numbers and hyphens. - Can not be "admin" - No underscores	`logout`	Otherwise, display an error message. User gets taken back to the login scope and may attempt to create an account again.
5 (11)	Check that `new_acc_holder` is unique; hasn't been used before.		If `new_acc_holder` is valid, prompt user for `init_balance`. Otherwise, display an error message. User gets taken back to the login scope and may attempt to create an account again.

6
(12)
	Check that `init_balance` is a valid amount.
	- No more than 99999.99
		`login`
		`admin`
		create
		new_acc_holder
		init_balance
		`logout`
			If `init_balance` is valid, create new account. Inform the user they have successfully created an account and save the information provided to the bank account transaction file.
			
			Otherwise, display an error message. User gets taken back to the login scope and may attempt to create an account again.

Test Case Name: Delete

Description: Delete a bank account. Privileged transaction.

#	OBJECTIVE	INPUT **These are templates for valid input only**		EXPECTED OUTPUT

1
(13)
	Check that the command is `delete`.	`login`
		`admin`
		delete
		acc_holder
		acc_number
		yes
		`logout`
			`login`
			`admin`
			delete
			acc_holder
			acc_number
			no
			`logout`
				If command is `delete`, continue.
				
				If it is not an existing command, show error message. User may use the `help` command.

2
(14)
	Check that user is logged in.	`login`
		`admin`
		delete
		acc_holder
		acc_number
		yes
		`logout`
			`login`
			`admin`
			delete
			acc_holder
			acc_number
			no
			`logout`
				If user is logged in, continue.
				
				Otherwise, display an error message.

3 (15)	Check that current logged in user is an administrator.	`login` `admin` **delete** **acc_holder** **acc_number**	`login` `admin` **delete** **acc_holder** **acc_number**	If account mode is `admin`, prompt user for `acc_holder`. Otherwise, display an error message.

#	OBJECTIVE	INPUT		EXPECTED OUTPUT
		yes logout	no logout	
4 (16)	Check that `acc_holder` exists in the Bank System.	login admin **delete** **acc_holder** **acc_number** **yes** logout	login admin **delete** **acc_holder** **acc_number** **no** logout	If `acc_holder` exists, prompt user for `acc_number`. Otherwise, display an error message. User gets taken back to the login scope and may attempt to delete an account again.
5 (17)	Check that `acc_number` corresponds to `acc_holder`.	login admin **delete** **acc_holder** **acc_number** **yes** logout	login admin **delete** **acc_holder** **acc_number** **no** logout	If they match, prompt user for deletion confirmation prompt. Otherwise, display an error message. User gets taken back to the login scope and may attempt to delete an account again.
6 (18)	Check if the input for the confirmation prompt is valid (`yes`/`y` or `no`/`n`)	login admin **delete** **acc_holder** **acc_number** **yes** logout	login admin **delete** **acc_holder** **acc_number** **no** logout	If `yes` or `y`, then account is deleted successfully and the user is informed of this. save the information provided to the bank account transaction file. If `no` or `n`, then the deletion process gets cancelled. User gets taken back to the login scope and may attempt to delete an account again. If none of the above, display an error message stating the 4 options available. User gets taken back to the login scope and may attempt to delete an account again.

Test Case Name: Disable

Description: Disable a bank account. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT
1 (19)	Check that the command is `disable`.	login admin **disable** **acc_holder** **acc_number** logout	If command is `disable`, continue. If it is not an existing command, show error message. User may use the `help` command.
2 (20)	Check that user is logged in.	login admin **disable** **acc_holder**	If user is logged in, continue. Otherwise, display an error message.

#	OBJECTIVE	INPUT	EXPECTED OUTPUT
		`acc_number` `logout`	

3
(21)
	Check that current logged in user is an administrator.	`login`
		`admin`
		`disable`
		`acc_holder`
		`acc_number`
		`logout`
			If account mode is `admin`, prompt user for `acc_holder`.
			
			Otherwise, display an error message.

4
(22)
	Check that `acc_holder` exists in the Bank System.	`login`
		`admin`
		`disable`
		`acc_holder`
		`acc_number`
		`logout`
			If `acc_holder` exists, continue.
			
			Otherwise, display an error message. User gets taken back to the login scope and may attempt to disable an account again.

5 (23)	Check that the account hasn't already been disabled.		If it is not disabled, prompt user for `acc_number`. Otherwise, display an error message. User gets taken back to the login scope and may attempt to disable an account again.

6
(24)
	Check that `acc_number` corresponds to `acc_holder`.	`login`
		`admin`
		`disable`
		`acc_holder`
		`acc_number`
		`logout`
			If they match,then account is disabled successfully and the user is informed of this. Save the information provided to the bank account transaction file *immediately*. Change bank account from active (A) to disabled (D).
			
			Otherwise, display an error message and abort disablement. User is taken back to the login scope and may attempt to disable an account again.

Test Case Name: Enable

Description: Enable a bank account. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT

1
(26)
	Check that the command is `enable`.	`login`
		`admin`
		`enable`
		`acc_holder`
		`acc_number`
		`logout`
			If command is `enable`, continue.
			
			If it is not an existing command, show error message. User may use the `help` command.

2	Check that user is	`login`	If user is logged in, continue.

(27)	logged in.	admin **enable** **acc_holder** **acc_number** logout	Otherwise, display an error message.
3 (28)	Check that current logged in user is an administrator.	login admin **enable** **acc_holder** **acc_number** logout	If account mode is admin, prompt user for acc_holder. Otherwise, display an error message.
4 (29)	Check that acc_holder exists in the Bank System.	login admin **enable** **acc_holder** **acc_number** logout	If acc_holder exists, then continue. Otherwise, display an error message. User gets taken back to the login scope and may attempt to enable an account again.
5 (30)	Check that the account is actually disabled.		If the account is disabled, then prompt user for acc_number. Otherwise, display an error message. User gets taken back to the login scope and may attempt to enable an account again.
6 (31)	Check that acc_number corresponds to acc_holder.	login admin **enable** **acc_holder** **acc_number** logout	If they match, then the account is enabled successfully and the user is informed of this. Save the information provided to the bank account transaction file *immediately*. Change bank account from disabled (D) to active (A). Otherwise, display an error message and abort enablement. User is taken back to the login scope.

Test Case Name: Change Plan
Description: Change the transaction payment plan for an account. Privileged transaction.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT
1 (33)	Check that the command is changeplan.	login admin **changeplan** **acc_holder** **acc_number**	If command is changeplan, continue. If it is not an existing command, show error message. User may use the help command.

#	OBJECTIVE	INPUT	EXPECTED OUTPUT
		`logout`	

2 (34)	Check that user is logged in.	`login`
		`admin`
		changeplan
		acc_holder
		acc_number
		`logout`
			If user is logged in, continue.
			
			Otherwise, display an error message.

3 (35)	Check that current logged in user is an administrator.	`login`
		`admin`
		changeplan
		acc_holder
		acc_number
		`logout`
			If account mode is `admin`, prompt user for `acc_holder`.
			
			Otherwise, display an error message.

4 (36)	Check that `acc_holder` exists in the Bank System.	`login`
		`admin`
		changeplan
		acc_holder
		acc_number
		`logout`
			If `acc_holder` exists, then continue.
			
			Otherwise, display an error message. User gets taken back to the login scope and may attempt to change plans on an account again.

	Check that `acc_holder` is enabled/active.		If acc_holder is enabled/active, then prompt user for `acc_number`. Otherwise, display an error message. User gets taken back to the login scope and may attempt to change plans on an account again.

5 (37)	Check that `acc_number` corresponds to `acc_holder`.	`login`
		`admin`
		changeplan
		acc_holder
		acc_number
		`logout`
			If they match, changes plans successfully and the user is informed of this. Save the information provided to the bank account transaction file *immediately*. Change account payment plan from non-student (N - default) to student (S) or vice versa.
			
			Otherwise, display an error message. User gets taken back to the login scope and may attempt to change plans on an account again.

Test Case Name: Deposit

Description: Deposit money into an account.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT
1 (38)	Check that the command is `deposit`.	`login` `admin`	`login` `standard`	If command is `deposit`, continue.

		deposit **acc_holder** **acc_number** **amount_dep** logout	acc_holder **deposit** **acc_number** **amount_dep** logout	If it is not an existing command, show error message. User may use the help command.
2 (39)	Check that user is logged in.	login admin **deposit** **acc_holder** **acc_number** **amount_dep** logout	login standard acc_holder **deposit** **acc_number** **amount_dep** logout	If user is logged in, continue. Otherwise, display an error message.
3 (40)	Check account mode (admin or standard)	login admin **deposit** **acc_holder** **acc_number** **amount_dep** logout	login standard acc_holder **deposit** **acc_number** **amount_dep** logout	If account mode is admin, prompt user for acc_holder. if account mode is standard, skip acc_holder and prompt user for acc_number. Otherwise, display an error message.
4 (41)	Check that acc_holder exists in the Bank System.	login admin **deposit** **acc_holder** **acc_number** **amount_dep** logout	*Doesn't apply.*	If acc_holder exists, then continue. Otherwise, display an error message. User gets taken back to the login scope and may attempt to deposit funds into an account again.
	Check that the account is enabled/active		login standard acc_holder **deposit** **acc_number** **amount_dep** logout	If the account is enabled/active, then continue. Otherwise, display an error message. User gets taken back to the login scope and may attempt to deposit funds into an account again.
	Check that the account was not created in the same day.			If the account was not created in the same day, then prompt user for acc_number. Otherwise, display an error message. User gets taken back to the login scope and may attempt to deposit funds into an account again.
5 (42)	For administrators, check that acc_number corresponds to acc_holder. For standard users, check that the acc_number	login admin **deposit** **acc_holder** **acc_number** **amount_dep** logout	login standard acc_holder **deposit** **acc_number** **amount_dep** logout	If they match, then account changes plans successfully and the user is informed of this. Save the information provided to the bank account transaction file. Change account payment plan from student (SP) to non-student (NP) or vice versa.

#	OBJECTIVE	INPUT		EXPECTED OUTPUT
	corresponds to the currently logged in `acc_holder`.			Otherwise, display an error message. User gets taken back to the login scope and may attempt to deposit funds into an account again.
8 (45)	Check that `amount_dep` is valid. - Must be in the form of #####.##. No dollar (or any other) signs. 99999.99 - Only positive numerical values *bigger* than 0.	login admin **deposit** **acc_holder** **acc_number** **amount_dep** logout	login standard acc_holder **deposit** **acc_number** **amount_dep** logout	If `amount_dep` is valid, then continue. Otherwise, display an error message. User gets taken back to the login scope and may attempt to deposit funds into an account again.
9 (46)	Check that the total amount to be withdrawn is valid. - For standard users: `amount_dep` + balance - fee <= 99999.99 - For administrators: `amount_dep` <=			If the total amount to be deposited is valid, then the amount specified is deposited into the account and the user is informed of this. Save the information provided to the bank account transaction file. In the case of a standard user *only*, a deposit fee is deducted. Otherwise, display an error message. User gets taken back to the login scope and may attempt to deposit funds into an account again.

Test Case Name: Withdrawal

Description: Withdraws money from an account.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT
1 (47)	Check that the command is `withdrawal`.	login admin **withdrawal** **acc_holder** **acc_number** **amount_wit** logout	login standard acc_holder **withdrawal** **acc_number** **amount_wit** logout	If command is `withdrawal`, continue. If it is not an existing command, show error message. User may use the `help` command.
2 (48)	Check that user is logged in.	login admin **withdrawal** **acc_holder** **acc_number** **amount_wit** logout	login standard acc_holder **withdrawal** **acc_number** **amount_wit** logout	If user is logged in, continue. Otherwise, display an error message.

3 (49)	Check account mode (admin or standard)	login admin **withdrawal** **acc_holder** **acc_number** **amount_wit** logout	login standard acc_holder **withdrawal** **acc_number** **amount_wit** logout	If account mode is admin, prompt user for acc_holder. if account mode is standard, skip acc_holder and prompt user for acc_number. Otherwise, display an error message.
4 (50)	Check that acc_holder exists in the Bank System.	login admin **withdrawal** **acc_holder** **acc_number** **amount_wit** logout	*Doesn't apply.*	If acc_holder exists, then continue. Otherwise, display an error message. User gets taken back to the login scope and may attempt to transfer funds to an account again.
	Check that the account is enabled/active.		login standard acc_holder **withdrawal** **acc_number** **amount_wit** logout	If the account is enabled/active, then continue. Otherwise, display an error message. Withdrawal gets aborted.
	Check that the account was not created in this same day.			If the account is was not created in this same day, prompt user for acc_number. Otherwise, display an error message. Withdrawal gets aborted.
5 (51)	For administrators, check that acc_number corresponds to acc_holder. For standard users, check that the acc_number corresponds to the currently logged in acc_holder.	login admin **withdrawal** **acc_holder** **acc_number** **amount_wit** logout	login standard acc_holder **withdrawal** **acc_number** **amount_wit** logout	If they match, then the user is prompted for amout_wit. Otherwise, display an error message. User gets taken back to the login scope and may attempt to transfer funds to an account again.
8 (54)	Check that amount_wit is valid. - Must be in the form of #####.00. No dollar (or any other) signs. - No decimals (pennies) - Multiple of 5 - Only positive numerical values *bigger* than 0. - No funds that were deposited in this same day.	login admin **withdrawal** **acc_holder** **acc_number** **amount_wit** logout	login standard acc_holder **withdrawal** **acc_number** **amount_wit** logout	If amount_wit is valid, then continue. Otherwise, display an error message. User gets taken back to the login scope and may attempt to transfer funds to an account again.

#	OBJECTIVE			EXPECTED OUTPUT

| 9 (55) | Check that the total amount to be withdrawn is valid. - `amount_wit` + fee <= balance | | | If the total is valid, then the amount specified is withdrawn from the account and the user is informed of this. Save the information provided to the bank account transaction file. Otherwise, display an error message. User gets taken back to the login scope and may attempt to transfer funds to an account again. |

Test Case Name: Transfer

Description: Transfer money between two accounts.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT
1 (56)	Check that the command is `transfer`.	login admin **transfer** **acc_holder** **acc_num_f** **acc_num_t** **amount_tra** logout	login standard acc_holder **transfer** **acc_num_f** **acc_num_t** **amount_tra** logout	If command is `transfer`, continue. If it is not an existing command, show error message. User may use the `help` command.
2 (57)	Check that user is logged in.	login admin **transfer** **acc_holder** **acc_num_f** **acc_num_t** **amount_tra** logout	login standard acc_holder **transfer** **acc_num_f** **acc_num_t** **amount_tra** logout	If user is logged in, continue. Otherwise, display an error message.
3 (58)	Check account mode (`admin` or `standard`)	login admin **transfer** **acc_holder** **acc_num_f** **acc_num_t** **amount_tra** logout	login standard acc_holder **transfer** **acc_num_f** **acc_num_t** **amount_tra** logout	If account mode is `admin`, prompt user for `acc_holder`. if account mode is standard, skip `acc_holder` and prompt user for `acc_number`. Otherwise, display an error message.
4 (59)	Check that `acc_holder` exists in the Bank System.	login admin **transfer** **acc_holder** **acc_num_f** **acc_num_t** **amount_tra**	*Doesn't apply.*	If `acc_holder` exists, prompt user for `acc_num_f`. Otherwise, display an error message. Prompt user for `acc_holder` again.
	Check that the current		login	If the current account is

#	Check	(admin)	(standard)	Result

	account is enabled/active.	`logout`	`standard`
			`acc_holder`
			`transfer`
			`acc_num_f`
			`acc_num_t`
			`amount_tra`
			`logout`
				enabled/active, continue.
				
				Otherwise, display an error message. Transfer gets aborted.

	Check that the current account was not created in this same day.			If the current account was not created in this same day, then continue. Otherwise, display an error message. Transfer gets aborted.

5 (60)	For administrators, check that `acc_num_f` corresponds to `acc_holder`.
	
	For standard users, check that the `acc_num_f` corresponds to the currently logged in `acc_holder`.
		`login`
		`admin`
		`transfer`
		`acc_holder`
		`acc_num_f`
		`acc_num_t`
		`amount_tra`
		`logout`
			`login`
			`standard`
			`acc_holder`
			`transfer`
			`acc_num_f`
			`acc_num_t`
			`amount_tra`
			`logout`
				If they match, then the user is prompted for `acc_num_t`.
				
				Otherwise, display an error message. Prompt user for `acc_num_f` again.

8 (63)	Check that `acc_num_t` exists in the Bank System.	`login`
		`admin`
		`transfer`
		`acc_holder`
		`acc_num_f`
		`acc_num_t`
		`amount_tra`
		`logout`
			`login`
			`standard`
			`acc_holder`
			`transfer`
			`acc_num_f`
			`acc_num_t`
			`amount_tra`
			`logout`
				If `acc_num_t` exists, then continue.
				
				Otherwise, display an error message. Prompt user for `acc_num_t` again.

9 (64)	Check that the destination account is enabled/active.			If the destination account is enabled/active, then continue. Otherwise, display an error message. Transfer gets aborted.
10 (65)	Check that the destination account was not created in this same day.			If it was not created in this same day, then prompt the user for `amount_tra`. Otherwise, display an error message. Transaction gets aborted.

11 (66)	Check that `amount_tra` is valid.
	- Must be in the form of #####.##. No dollar (or any other) signs.
	- No more than 1000.00 for current session (for standard users only) 1000.00 <= total transferred in this session + `amount_tra`
	- Only positive numerical values
		`login`
		`admin`
		`transfer`
		`acc_holder`
		`acc_num_f`
		`acc_num_t`
		`amount_tra`
		`logout`
			`login`
			`standard`
			`acc_holder`
			`transfer`
			`acc_num_f`
			`acc_num_t`
			`amount_tra`
			`logout`
				If `amount_tra` is valid, then continue.
				
				Otherwise, display an error message. Prompt user for `amount_tra` again.

#	OBJECTIVE			EXPECTED OUTPUT
	bigger than 0. - No funds that were deposited in this same day			
12 (67)	Check that the total amount to be transferred is valid. - amount_tra + fee <= balance			If the total is valid, then the amount specified is transferred from acc_num_f into acc_num_t and the user is informed of this. Save the information provided to the bank account transaction file. In the case of a standard user *only*, a transfer fee is deducted. Otherwise, display an error message. Prompt user for amount_tra again.

Test Case Name: Paybill

Description: Pay bill with money from an account.

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**		EXPECTED OUTPUT
1 (68)	Check that the command is paybill.	login admin **paybill** **acc_holder** **acc_num** **company** **amount_pay** logout	login standard acc_holder **paybill** **acc_num** **company** **amount_pay** logout	If command is paybill, continue. If it is not an existing command, show error message. User may use the help command.
2 (69)	Check that user is logged in.	login admin **paybill** **acc_holder** **acc_num** **company** **amount_pay** logout	login standard acc_holder **paybill** **acc_num** **company** **amount_pay** logout	If user is logged in, continue. Otherwise, display an error message.
3 (70)	Check if account mode (admin or standard) is valid.	login admin **paybill** **acc_holder** **acc_num** **company** **amount_pay** logout	login standard acc_holder **paybill** **acc_num** **company** **amount_pay** logout	If account mode is admin, prompt user for acc_holder. If account mode is standard, skip acc_holder and prompt user for acc_number. Otherwise, display an error message.

4 (71)	Check that `acc_holder` exists in the Bank System.	login admin **paybill** **acc_holder** **acc_num** **company** **amount_pay** logout	*Doesn't apply.*	If `acc_holder` exists, then continue. Otherwise, display an error message. Prompt user for `acc_holder` again.
	Check that the account is enabled/active		login standard acc_holder **paybill** **acc_num** **company** **amount_pay** logout	If the account is enabled/active, then continue. Otherwise, display an error message. Bill payment gets aborted.
	Check that the account was not created in this same day.			If the account was not created in this same day, prompt user for `acc_num`. Otherwise, display an error message. Bill payment gets aborted.
5 (72)	For administrators, check that `acc_num` corresponds to `acc_holder`. For standard users, check that the `acc_num` corresponds to the currently logged in `acc_holder`.	login admin **paybill** **acc_holder** **acc_num** **company** **amount_pay** logout	login standard acc_holder **paybill** **acc_num** **company** **amount_pay** logout	If they match, then prompt the user for `company`. Otherwise, display an error message. Prompt user for `acc_num` again.
8 (75)	Check that `company` is valid. Must be one of the following: `EC`, `CQ`, or `TV`.	login admin **paybill** **acc_holder** **acc_num** **company** **amount_pay** logout	login standard acc_holder **paybill** **acc_num** **company** **amount_pay** logout	If `company` is valid, prompt the user for `amount_pay`. Otherwise, display an error message. Prompt user for `company` again.
9 (76)	Check that `amount_pay` is valid. - No more than 2000.00 (for standard users only, per day per company) 2000.00 <= total paid in this same day + `amount_pay` - Only positive numerical values - No funds that were deposited this same day.	login admin **paybill** **acc_holder** **acc_num** **company** **amount_pay** logout	login standard acc_holder **paybill** **acc_num** **company** **amount_pay** logout	If `amount_pay` is valid, then continue. Otherwise, display an error message. Prompt user for `amount_pay` again.
10 (77)	Check that the total amount to be paid is valid.			If the total is valid, then the amount specified is paid to the corresponding `company` and the user

#			
	- amount_pay + fee <= balance			is informed of this. Save the information provided to the bank account transaction file. Otherwise, display an error message. Prompt user for amount_pay again.

Test Case Name: Help

Description: Displays all available commands with their corresponding descriptions and arguments

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT
1 (78)	Check if command is help	**help**	If command is help, display command details. If it is not an existing command, show error message. User may use the help command.

Test Case Name: Quit

Description: Terminates the program

#	OBJECTIVE	INPUT **Will only show all options when relevant to the process**	EXPECTED OUTPUT
1 (79)	Check if command is quit	**quit**	If command is quit, continue. If it is not an existing command, show error message. User may use the help command.
2 (80)	Check if user is currently logged in		If not logged in, then terminate the system. Otherwise, display an error message.