ATM Analysis

**Problem Statement:**

*The program must:*

-Scan a .txt file for data, and read the data.

-Allow user to enter a username and password.

-Verify that the username and password is correct.

-Use defensive programming to ensure the user enters the correct inputs.

-Allow the user to choose whether they want to access savings or checking.

-Give the user the option to choose between 5 possible transactions.

-Terminate after 3 transactions.

-Be user friendly

-Perform calculations to add and subtract funds (deposit/withdrawal/transfer)

**Problem Analysis:**

The program will begin by scanning a .txt file and reading the data. The user will then be prompted for a username, only three attempts are allowed before an error message appears prompting the user to restart the program. If the username is correct, they will be prompted for a password, again- only three attempts are allowed before an error message appears. If the user has the correct user/pass a welcome message is displayed, followed by three transactions. The user will have the option of choosing between accessing their savings account, or checking account. Then a menu will appear with five possible transaction choices: deposit, withdrawal, balance inquiry, transfer funds, or exit.

Deposit: Prompts user for a positive integer amount. Then adds amount to [saving; checking]

Withdrawal: Prompts user for a positive integer amount. Then subtracts amount from [saving; checking]

Balance Inquiry: Displays current balance of [saving; checking]

Transfer Funds: Prompts user for a positive integer. Then adds amount to [saving; checking] and subtracts amount from [saving; checking].

Exit: Terminates program.

At the end of the program goodbye message will appear.

(Defensive programming is used to ensure the user enters the correct input).

**Program Design:**

Call Get\_Doc\_Info

Set Count to 1

Read Text Document

Repeat Until End\_Of\_Input

Get Customer[count]

Get Username[count]

Get Password[count]

Get Saving[count]

Get Checking[count]

Set count to count + 1

End Repeat

Call Verify\_Username

Set Attempt to 3

Print “Please enter username.”

Repeat Until username == username[count]

Input Username

IF username == username[count] THEN

Set index to count

ELSE Print “Wrong username, please try again.”

Set Attempt to Attempt - 1

IF Attempt == 0 THEN terminate

End Repeat

Call Verify\_Password

Set Attempt to 3

Print “Please enter password.”

Repeat Until password == password[index]

Input Password

IF password == password[index] [continue]

ELSE Print “Wrong password, please try again.”

Set Attempt to Attempt - 1

IF Attempt == 0 THEN terminate

End Repeat

Call Selection

Print “Welcome “ + Customer[index]

Repeat Until Transaction == 0 OR Choice == 5

Set Transaction to 4

Set Transaction to Transaction – 1

Print Transaction

Print “For savings, enter 1. For checking, enter 0.”

Get Account

IF Account == 1 THEN

Print “To deposit, enter 1. Withdrawal, enter 2. Balance Inquiry, enter 3. Transfer Funds, enter 4. Exit, enter 5.”

Get Choice

IF Choice == 1 call Saving\_Deposit

If Choice == 2 call Saving\_Withdrawal

If Choice == 3 call Saving\_Balance\_Inquiry

If Choice == 4 call Saving\_Transfer

If Choice == 5 terminate ELSE

Print “To deposit, enter 1. Withdrawal, enter 2. Balance Inquiry, enter 3. Transfer Funds, enter 4. Exit, enter 5.”

IF Choice == 1 call Checking\_Deposit

If Choice == 2 call Checking\_Withdrawal

If Choice == 3 call Checking\_Balance\_Inquiry

If Choice == 4 call Checking\_Transfer

If Choice == 5 terminate

End repeat

Print “Thank you for using the Ivy Tech ATM!”

**Program Code:** [See corbin\_caitlin\_ATM\_Final]

**Program Test:** [See corbin\_caitlin\_ATM\_Final]

*Discuss your approach to securing your code from invalid data.*

**Defensive Programming:**

If you are wanting an input that is a number, use Is\_Number(variable). If you are wanting to ensure the number is an integer, use a variation on floor(variable). If you are searching for a certain ranger of numbers, use relational operators. Use loops and selection structures for defensive programming as well.

*Create and document test data to ensure it is error free.*

**Error Testing:**

-No spelling errors

-Arithmetic is correct

-Data in text file is stored in parallel arrays

-Defensive programming only allows positive integer numbers

-Error message displays after allowed attempts are used up

-Program terminates after 3 transactions, or if user exits

-All customer info has been tested