CSC 401 ASSIGNMENT SEVEN

Due Date: Tuesday, Oct. 31st by 11:58 PM

The purpose of this assignment is to assess your understanding of

- Encapsulation in functions
- Namespaces: global vs local
- Exception handling
- Program modularity

SUBMISSION

- Include your full name as a comment on the first line of your Python program.
- Code this problem in one Python file(.py) as YourName_HW.py
- Upload this file to Submissions folder
- Late assignments are not accepted. To earn partial credit you must submit the work you completed by the deadline.

PROBLEM

Note: You may not use Python statements, functions, data type, etc. that were not discussed in the reading assignment or the lecture notes/videos for this week or previous weeks. This is a class for students who have not programmed before and I expect everyone to code on the same level. If you have a better way of writing the code, then upload two versions: one that codes according to the specifications and the other that demonstrates advanced programming techniques.

This program gives you the opportunity to use all the statements and concepts we have discussed so far in the course plus exception handling. All variables must be local and passed into and from functions**. There should a return statement in each function, even if nothing is returned.**

REQUIREMENTS

You are to simulate a simple ATM machine. An account holder (user) should be able to enter their pin number and select from a menu of transactions: Deposit, Withdraw, Balance or Quit. You are to assume that the user has only one account on which these transactions can be performed. This account is associated with the user's pin number. The ATM should properly and regularly communicate with the user. The ATM should get information on current account information from a file. An **accounts.csv** file is posted with this assignment. Examine the structure of each line of the file which contains a 4-digit code, first and last names, and the current account balance.

Note: whenever I use **'Code the exception'**, this means you need to include code in a 'try' block to catch the exception caused by a runtime error.

You **may not** use global variables; all variables must be local. You **may not** change the names of the functions or the parameters that are passed.

1. (10 pts.)Write a function **main()** that will control the flow of your program. You will build this code as you continue to create functions.

    - **main()** should call **startup**, **getUser** and **menu.**

    - It should contain a **loop** that determines which transaction the user has chosen to process and allows the user to process as many transactions as desired.

        - If the user chooses to make a deposit, the **deposit** function is called and the dictionary is updated with the returned balance.
        - If the user chooses to make a withdrawal, the **withdraw** function is called and the dictionary is updated with the returned balance.
        - If the user chooses to check the account balance, the **balance** function is called.
        - If the user chooses to quit, exit the loop and print ('Goodbye'). Once, the user quits, the program should terminate. Note: suggestion - use a Boolean data type to determine when the loop should terminate

2. (10 pts.)Write a function **startUp(fname)** that takes as input a filename that contains the account holder information and current balance. The file should be read and the data stored in a **dictionary** with the pin as the key and the remaining information as the value. Note: the current balance is a float data type. Do an explicit conversion to float on that list item. **Code an exception** ('Cannot get to the file').
   The function should **return** two values: Boolean, dictionary; for example: if you named your dictionary, daccts, and the file was successfully read and the dictionary successfully created, then **return (True, daccts)**; if the file was not successfully read, **return (False, daccts)**.

3. (5 pts.)Write a function **getUser(proceed, dictionary)** that takes as input a Boolean value and the dictionary you created in **startUp**. Prompt the user to enter a pin number. If the pin is valid, the function should **return (user, proceed)**, where user is the dictionary key value; if the pin is invalid, print ('Incorrect pin') and **return (None, False)**.

4. (5 pts.) Write a function **menu(name)** that takes as input the user's first name, greets the user and displays the menu options: 1. Deposit, 2: Withdraw, 3.Check Balance, 4. Quit. If the user enters an invalid value (i.e. not 1, 2, 3 or 4) display the menu option again and give the user a chance to enter a value again. Once a valid number is entered, the function should **return t**he number of the chosen option.

5. (5 pts.)Write a function **deposit(balance)** that takes as input the user's account balance. The function calls getAmount(), calculates and displays the new balance. The function **returns the new balance**.

6. (10 pts.)Write a function **getAmount()** that takes no input. Inside a loop, prompt the user for an amount to be either deposited or withdrawn, at this point it does not matter which it is. The amount must be converted to float. **Code an exception** ('You entered an incorrect amount. Please try again'). An interruption would be caused if the user enters a string instead of a numeric value, or presses the enter key without entering anything. Stay in the loop until a valid number is entered and **return amount**.

7. (5 pts.)Write a function **withdraw(balance)** that takes as input the user's account balance. The function calls **getAmount().** If the amount to be withdrawn is greater than the balance display ('Insufficient funds to complete the transaction'). The user should be prompted to enter a new amount, until the amount entered is less than the balance. The function calculates and displays the new balance. The function **returns the new balance**.

8. (5 pts.) Write a function **balance(name, balance)** that takes as input the user's name and balance and prints the message 'Your current balance is $xxxx.xx' where xxxx.xx is the current balance amount. This function **does not return a value**.

The following is only one test case. You should do several test cases to evaluate all the exceptions you have coded and other possible data entry errors.
Sample execution:

```
Welcome -- Please enter your pin number 4466

Kay :
1: Deposit
2: Withdrawal
3: Check Balance
4: Quit

Enter number: 3
Your current balance is $356.00

Kay :
1: Deposit
2: Withdrawal
3: Check Balance
4: Quit

Enter number: 2
Amount: 600
Insufficient funds to complete the transaction
Amount: 300
Your new balance is $56.00

Kay :
1: Deposit
2: Withdrawal
3: Check Balance
4: Quit

Enter number: 1
Amount: 100
Your new balance is $156.00

Kay :
1: Deposit
2: Withdrawal
3: Check Balance
4: Quit

Enter number: 4
Goodbye
```

IF YOU HAVE ANY QUESTIONS REGARDING THIS ASSIGNMENT, PLEASE PORT THEM TO THE ASSIGNMENT SEVEN DISCUSSION FORUM