## Programming Problem 5 - bank.py

Directions: Download the template files I have provided on Blackboard. Then open Spyder, load the template file, and write the following program. Submit your source code via Gradescope in `.py` format; do NOT send any other files. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

**Be sure to read the SPECIFICATIONS carefully! And write comments!**

For this problem, you are tasked with the creating a new class called `Bank`. Original creation of the bank object will take one argument called `name` and each object begins with two attributes, `_name` and `_accounts`. The `_name` attribute will be declared by the user upon creation and the `_accounts` attribute will begin as an empty dictionary. For example

```
bigbank = Bank('BigBank')
print(bigbank._name)
print(bigbank._accounts)
```

will print `BigBank` and `{}` (an empty dictionary) on two lines.

Your goal is to define the basic functionality of a bank. That is, you should be able to create accounts, display account balance, make deposits, make withdrawals, and make transfers (between accounts owned by the same user).

You're going to do by writing several methods:

1. `createaccount` which takes three parameters: customer name, initial checking, initial savings. When the user calls this method, it should add an entry to the `_accounts` dictionary with key defined by the customer name and the value defined as a dictionary containing the checking and savings account balance. The **default values** for checking and savings account balances should be 0 if they are not specified. You may assume that the user is **using unique customer names** so no duplicate accounts are created.
2. `display` which takes one parameter: customer name. This method should print the account balance of all accounts in that users name.
3. `deposit` which takes two parameters; customer name, value. This method should make a deposit to the users checking account. This method should not allow negative deposits! If the user attempts to do so, they should be met with an appropriate message.
4. `withdrawal` which takes two parameters; customer name, value. This method should make a withdrawal from the customers checking account. This method should not allow negative withdrawals or withdrawals that will overdraft the checking leaving the checking account with a negative balance. If the user attempts to do so, they should be met with an appropriate message.
5. `transfer` which takes four parameters: customer name, initial account, destination account, amount to transfer. This method should make a transfer of the desired amount from the initial account to the destination. This transfer should not be completed unless the customer has enough money in their initial account to make the transfer. The **default** transfer should move from checking to savings.

Be sure to include docstrings for each of the methods to describe what each method does.

---

**Specifications**: your program must

- contain methods 1-5 which behave as described.
- contain docstrings for each method written, which describe the relationship between arguments and return values.

**There is no autograder for this assignment!** You should test your class creation thoroughly before submission. I will pretending to be a user who creates 4 accounts and tests each of your methods.