CS995: Assignment

1. Write a class called `ItemAndQty`. This class should have public data members to hold a name, a price and quantity. The name is a string, the price is a floating point number and the quantity is an integer (or whole) number.
2. Write a `__repr__` function for the `ItemAndQty` class that returns its data members within a string, together with their data member names.
3. Write a `cost` function for the `ItemAndQty` class that returns a value from the multiplication of the price and quantity data members.
4. Write a class called `Shop`. The class should have one data member, which is a dictionary of the name of an `ItemAndQty` object and an `ItemAndQty` object. The key of the dictionary should be the name of the `ItemAndQty` object. The value of the dictionary should be the `ItemAndQty` object.
5. Write an `addItemAndQty` function for the `Shop` class. This function should accept an item name, price and quantity. It should create an `ItemAndQty` object if one does not already exist in the `Shop` object with the same name. If one already exists, then the quantity of the `ItemAndQty` object in the `Shop` object should be incremented by the quantity supplied.
6. Write a `loadInitialStock` function for the `Shop` class. This function should read a CSV file that contains columns of item name, price and quantity and create `ItemAndQty` objects in the `Shop` object by calling the `addItemAndQty` function.
7. Write an `itemAndQtyByName` function for the `Shop` class. This function should be passed the item name and return an `ItemAndQty` object if one exists in the `Shop` object. If no object exists by the supplied item name, the function should return `None`.
8. Write an `itemsInStock` function for the `Shop` class. This function should be passed the item name and return the quantity of the item in the shop. This function should call the `itemAndQtyByName` function and then return the item quantity or zero if the object is not in the `Shop` object.
9. Write a class called `ShoppingBasket`. The `ShoppingBasket` class constructor should accept a `Shop` object and create an empty dictionary of the name of an `ItemAndQty` object and an `ItemAndQty` object. The key of the dictionary should be the name of the `ItemAndQty` object. The value of the dictionary should be the `ItemAndQty` object.
10. Write an `addItemAndQty` function for the `ShoppingBasket` class. The function should accept an item name and quantity. The function should:
    - Check if the item is in the `Shop` object. If it is not, then it should return zero.
    - Add the quantity of `ItemAndQty` that are available in the `Shop` object, up to the number requested to be added. The function should use the price of the `ItemAndQty` in the `Shop` object to do this.
    - Reduce the quantity in the `Shop` object by the number that are added.
    - Return the number of items that are added.

11. Write a `totalCost` function for the `ShoppingBasket` class. The function should return the total cost of all of the items in the shopping basket, by calling the `cost` function of each of the `ItemAndQty` objects.

12. Write an `empty` function for the `ShoppingBasket` class. The function should clear the data member dictionary, such that the shopping basket is empty.

13. Write a unit test class. The unit test class should contain functions to:

    - Create a `Shop` object. Test the loading of the CSV data into the `Shop` object by creating a CSV file and then calling the `Shop` class `loadInitialStock` member function. The CSV file can be created using a text editor or spreadsheet application.
    - Create a `Shop` object. Add an item to the shop by calling the `Shop` member function `addItemAndQty`. Then test the response of the `Shop` member function `itemsInStock` for the added item and an item that does not exist in the `Shop` object.
    - Create a `Shop` object. Add several items to the `Shop` object by calling the `Shop` member function `addItemAndQty`. Test the `ShoppingBasket` member function `addItemAndQty` and `totalCost`. Then test the `Shop` member function `itemsInStock` returns the number of items that are left in stock.

All files that are written to address these steps should be added to a zip file, which is then submitted through the CS995 MyPlace page. The zip file should include Python files and the CSV file needed to text the `loadInitialStock` function.