STSCI 4060 HW3

(Assigned on: 3/5/2019, Due: 11:59 PM, 3/12/2019)

General Instruction

- **Do your own work. Cornell academic integrity rules are enforced. Copying code from the Internet, fellow students or elsewhere is prohibited and will result in a penalty of at least getting 0 points for the entire assignment.**

- **You must include some comments to show the question number and what your program does for each Python program. If this part is missing, up to 5 points will be deducted.**

- **What to submit:**
  1. All the Python files, including grocery.py, linkedList.py, linkedListTest.py, and the Node class module (refer to the lecture notes), etc.
  2. HW3Q1_outputs.txt.
  3. An MS Word (or PDF) file, containing a screenshot of the outputs from linkedListTest.py.
- **How to submit:** Upload a single zipped file, for example, STSCI4060HW3_LastName_FirstName.7z, containing all the necessary components to Blackboard.

Questions

1. (50 points total) Define a class call Grocery, which contains a constructor and three methods: buildInventory(), viewInventory() and checkOut(), to handle a small online grocery store's inventory operations and checkout transactions. When displaying/printing something, use a format string whenever possible. Name this Python program grocery.py.

   A. (2 points) In the constructor, define an empty dictionary called inventory.
   B. (2 points) The buildInventory() method takes an argument of dictionary type to build or update the inventory dictionary. The keys of the dictionary are the names of grocery items (all in capital letters) and the values are expressed as a list, containing the quantity of a specific grocery item (at index 0) and the price of the item (at index 1).
   C. (2 points) The viewInventory() method is to display the whole inventory. If the inventory is empty, display "The inventory is empty. Put an order ASAP."
   D. (30 points) The checkOut() method provides an interactive interface for the customers to enter what they want to buy and the quantity they want to buy. At the end, it prints out an invoice for the whole transaction. In this method, you first define and initialize some variables and an empty dictionary that will be used within the method. Here, the dictionary is to hold the information of the name(s) of item(s) sold and the quantity, price and the subtotal of each item.

The transaction starts with a question to the customer, "Do you want to buy something? " If the customer enters any of the following: "Yes", "yes"," Y", or "y" (If the answer is "No" or anything else, just display "Goodbye.") then a further question is asked, " Enter an item to buy: ". The name entered (all in capital letters) must match a name (a key) in the inventory dictionary. Also, the item must be in stock. Otherwise, it prints out a message, "Wrong item name or out of stock." and then ask "Do you still want to buy something? ". If the answer is "Yes", "yes"," Y", or "y" and then go back the beginning of the transaction by asking " Enter an item to buy: ". If the name is correct and the item is in stock, then ask "How many do you want? ". Based on the quantity entered, check if the quantity in stock is enough. If not, display a message saying "Not enough in stock; we can only sell you X item(s)." where X represents the actual quantity in stock. After an item is sold, immediately update the quantity of the item in the inventory. After one item is processed, ask the customer again, "Do you still want to buy something? " It repeats the same process as the above till the customer does not want to buy any more items.

Now, it is time to print the invoice, which must have a title and list the name(s) of item(s) sold and the quantity, price and subtotal of each item (one item per line), as well as the total amount to pay (see the screenshot below for an example of output). Keep two decimal points for all the prices, subtotals and total. The last line of your invoice is "Goodbye." and the time stamp (year, month, date, and the time of checkout). Leave a blank line after that. (Hint: the time stamp can be obtained by calling the asctime() function in the time module.)

```
Item            Quantity          Price    Subtotal
-------------------------------------------------
CHEESE          50                4.44     222.00
PEAR            20                1.11     22.20
APPLE           50                2.22     111.00
-------------------------------------------------
Please pay $355.20

Goodbye.  Tue Mar  5 20:53:26 2019

>>>
```

After the Grocery class has been implemented, run the script of the class definition and then do the following in the Python shell interactively: (a to n, 1 point each)

a. Instantiate an object for the Grocery class.
b. View the inventory.
c. Enter 10 grocery items of your choice to the inventory dictionary, including their names, quantities, and prices.
d. View the inventory.
e. Do a checkout: select three items of your choice and their quantities (<= the quantities in stock) from your inventory.
f. View the inventory.

g. Do a checkout: This time the customer entered an item name that is not in the inventory and then decided not to buy anything.
h. Do a checkout: This time the customer entered a correct item name but the quantity wanted was greater than it is in stock. Just buy this item with the available quantity.
i. Do a checkout: This time the customer entered a correct item name but it is out of stock and then the customer decided not to buy anything.
j. Now, at least one of the items is out of stock, order some quantity of the item and update the inventory.
k. View the inventory.
l. Add the 11th grocery item of your choice to the inventory. You can specify any arbitrary quantity and price for this item.
m. View the inventory.
n. Save all the above outputs to a file called HW3Q1_outputs.txt as a part of submission.

2. (50 points total) We have learned how to implement a linked list. Python code for several methods were given to you in the class, for example, isEmpty(), size(), add(element) and search(element). Now your task is to add more methods to the LinkedList class so that the class is more powerful. Name this Python program linkedList.py.

   A. (3 points) Add a method called printList() so that you can print (i.e., display) all the values of the linked list elements from the first one to the last one.
   B. (13 points) Add a method called findAll(element) so that you can find how many times a given element appears in the linked list and at what position(es), zero-based (same below), the element appears. If the given element does not exist in the list, your method displays the following message: "element was not found." where "element" is the given element to find. For example, you want to find how many times 500 appears in a linked list (whose values are [600, 500, 500, 400, 500, 300, 200, 100, 500]). Your findAll(500) method should display: " 500 is found 4 time(s) at position(s) [1, 2, 4, 8]. "
   C. (8 points) Add a method called insert(position, element) to insert an element at the specified position.
   D. (8 points) Add a method called popAtIndex(position) to remove the element at the specified position and return the value of the popped list element.
   E. (3 points) Add a method called appendLast(element) to append a new element to the end of the linked list.

   Now write a Python program named linkedListTest.py to test all the methods (old and new) to show that your LinkedList class works correctly. For each test, add a comment to document what the test is, and output the result. For example, after inserting 666 to [100, 200, 300, 400] at Position 2, you should output " After inserting 666 to the list at position 2, the list becomes [100, 200, 666, 300, 400]." Your submission must contain all the necessary files or modules so that your Python code runs correctly. All the files must be in the same folder or directory. (15 points for this last part)