Purpose: The purpose of this homework is to gain practice writing classes and using objects and classes to solve problems.

Instructions: This assignment consists of 3 problems, worth a total of 30 points.

Because your homework file is submitted and tested electronically, the following are very important:

- Submit a file called `hw12.py` to the Homework 12 submission link on Gradescope by the due date deadline.
    - You don't have to submit any other files, just `hw12.py`.
- Your program should run without errors when we execute it using Python 3.

The following will result in a score reduction equal to a percentage of the total possible points:

- Bad coding style (missing documentation, meaningless variable names, etc.) (up to 30%)
- Breaking problem-specific constraints (up to 40%)

Documentation:
You don't need to write documentation for individual functions/methods in this assignment, but you DO need to write documentation for each class instead. Put this template just inside of the class definition for every class you write.

```
'''
Purpose: (What does an object of this class represent?)
Instance variables: (What are the instance variables for this class,
and what does each represent in a few words?)
Methods: (What methods does this class have, and what does each do in
a few words?)
'''
```

Your friend Riley has given you $100, and asked you to go to the local shopping mall to purchase them a new outfit for an upcoming party, and then spend the rest on ice cream for the party. You really like ice cream, though, so you've decided to purchase Riley the cheapest possible outfit and maximize the amount you can spend on ice cream. You've downloaded the stores' inventory lists in advance to compute the cheapest possible outfit.

Download the CSV files contained in `hw12files.zip` on Canvas, and place them in the same folder as your `hw12.py` file. Each CSV file represents one clothing store in the mall. The first line is column titles (which will always be Item Name, Item Price, and Item Category, in that order). The item category is always one of four values, representing where the clothing item is typically worn: Head (hats, earmuffs, sunglasses, etc.), Torso (shirts, blouses, coats, etc.), Legs (pants, skirts, kilts, etc.), and Feet (shoes, boots, sandals, etc.). You will need one item of each category in order to complete Riley's outfit.

The objective is to find the cheapest possible item for each category within the mall (they don't all have to be from the same store). To do this, you'll need to define two classes, and one standalone function that uses said classes.

Constraints:

- Do not import/use any Python modules.
- Don't use the input() function, as this will break our grading scripts.
- You may use any string or list method that is appropriate to solving this problem.

Problem A. (10 *points*) **Item Class**

First, we'll define a class to represent a single item available for purchase.

The Item class must have four instance variables:

- `name` is a string describing what the item is called
- `price` is a float describing the item's price in USD
- `category` is a string that describes where the item is worn. It must be one of four values: 'Head', 'Torso', 'Legs', or 'Feet'.
- `store` is a string that describes the name of the store where the item can be found.

The Item class should also have these methods:

- The constructor `__init__(self, csv_string, store)` takes in two parameters other than self: `csv_string` and `store`.
    - `csv_string` consists of a string which has the name, price, and category for the item in that order, separated by comma (you're not actually opening a CSV file here, that's in Part B - this is just a string that's formatted like a line from a CSV file).
        - You need to use string methods to break this string up so that you can use the information to initialize the `self.name`, `self.price`, and `self.category` instance variables.
        - Remember that the price needs to be converted to a float.
    - `store` is a string representing the store where the item can be found, and should be used to initialize the `self.store` instance variable.
- Override the `__str__` method so that it returns a string in the format
  `"<name> (<category>): $<price>"`
  (see example below)
- Override the < operator (`__lt__`) so that it takes in another `Item` object, and returns `True` if the left operand (self) has a lower price than the right operand (other), or `False` otherwise.
- You can also include getters and setters for the instance variables (this is good practice if you are planning on learning to program in Java or C++, but is not a requirement).

Examples:
Copy the following if __name__ == "__main__" block into your hw12.py file, and comment out tests for parts of the class you haven't implemented yet. The lines that have output include the expected value next to them as a comment.

```
if __name__ == '__main__':
    safety_pants = Item('Orange Safety Pants,34.66,Legs', 'Sparkles')
    print(safety_pants.name)  #Orange Safety Pants
    print(safety_pants.price)  #34.66
    print(type(safety_pants.price))  #<class 'float'>
    print(safety_pants.category)  #Legs
    print(safety_pants.store)  #Sparkles
    print(safety_pants)  #Orange Safety Pants (Legs): $34.66
    helm = Item('Iron Greathelm,20.23,Head', 'Blacksmith')
    print(helm)  #Iron Greathelm (Head): $20.23
    print(safety_pants < helm)  #False
    print(helm < safety_pants)  #True
```

Problem B. (10 *points*) **Store Class**

Next, we'll define the Store Class, representing the inventory of a given Store. Each Store will contain several Items (note that you should not use inheritance here since the relationship is that a Store has Items, not that the Store is an Item).

The Store class must have two instance variables:

- `name` is a string, representing the name of the store
- `items` is a list of `Item` objects, representing every item in the store's inventory

The Store class should also have these methods:

- The constructor `__init__(self, name, filename)` takes in two parameters other than self: `name` and `filename`.
    - `name` is a string, representing the name of the store,
    - `filename` is another string, representing the name of the CSV file containing the data on the store's inventory.

  The constructor must open the specified file, and for each line (except the first, which has the column titles), it must create an `Item` object using the data from that line, and append it to the `self.items` list. You can assume that the file actually exists: no need for a try-except block to handle errors. You can also assume that the CSV file will be formatted as specified in the introduction. Make sure to remove the newline character '\n' from each line before passing it to the Item constructor.
- Override the `__str__` method so that it returns a string containing the name of the store, followed by the string representation of each item in the `self.items` list, separated by newlines (see example below). **You must call the `str()` function on each `Item` object in your `self.items` list to do this, no copy-pasting the code from Item.__str__.**
- You can also include getters and setters for the instance variables (this is good practice if you are planning on learning to program in Java or C++, but is not a requirement).

Examples: Assumes that you are running hw12.py from the same folder as all of the CSV files in hw12files.zip, which can be found on Canvas. Note that the memory locations (`0x039D42F8`, etc.) for the Item objects within the .items list will not be the same on your computer.

```
if __name__ == '__main__':
    print()
    store1 = Store('Sparkles', 'sparkles.csv')
    print(store1.name) #Sparkles
```

```
    print(store1.items) #[<__main__.Item object at 0x039D42F8>,
<__main__.Item object at 0x039D42C8>, <__main__.Item object at
0x039D41A8>, <__main__.Item object at 0x039D4448>, <__main__.Item
object at 0x039D44A8>, <__main__.Item object at 0x039D44D8>]

    print()
    print(store1)
    #Sparkles
    #Pikachu Shirt (Torso): $34.85
    #Cyan Roller-Blades (Feet): $4.62
    #Orange Safety Pants (Legs): $34.66
    #Light-Up Sneakers (Feet): $31.89
    #Starry Blouse (Torso): $25.89
    #Bedazzled Monocle (Head): $11.37

    store2 = Store('Goth City', 'gothcity.csv')
    print()
    print(store2)
    #Goth City
    #Black Leather Kilt (Legs): $32.54
    #Armored Trenchcoat (Torso): $17.84
    #Spiky Earrings (Head): $10.29
    #Batman Mask (Head): $6.56
```

Problem C. (10 *points*) **Cheapest Outfit**

Write a function `cheap_outfit(store_list)` that takes in one parameter, a list of `Store` objects. This function should not be inside of either class.

`cheap_outfit` should find the cheapest item for each category across all stores in the `store_list` list, and return a dictionary.

- The dictionary should have four keys, which are the strings that represent each category of item: `'Head'`, `'Torso'`, `'Legs'`, and `'Feet'`.
- For each key, the value should be an `Item` object that represents the cheapest item of the given category across the stores in `store_list`.
- You may assume that there will be at least one item for each category in at least one of the Stores in `store_list`.
- You may assume that there will be no ties for the cheapest item in any given category.

Hints:

- The built-in `min()` function works based on the < operator, so if you implemented < for `Item` objects correctly, using `min()` on a list of `Item` objects will return the one with the lowest price.
- You don't need to write documentation for this problem, since this would be a function and not a class.

Examples : Assumes that you are running hw12.py from the same folder as all of the CSV files in hw12files.zip, which can be found on Canvas.

Note that the memory locations (`0x03407310`, etc.) for the Item objects within the dictionary will not be the same on your computer. We can't see the contents of the items in the dictionary directly, so we use a for loop to print out the key-value pairs.

```
if __name__ == '__main__':
    print()
    outfit1 = cheap_outfit([Store('Professional Wear',
'professionalwear.csv')])
    print(outfit1) #{'Head': <__main__.Item object at 0x03407310>,
'Torso': <__main__.Item object at 0x03407340>, 'Legs': <__main__.Item
object at 0x03407250>, 'Feet': <__main__.Item object at 0x034071F0>}
    print()
    for key in outfit1:
        print(key,' - ',outfit1[key])
```

```
    #Head  -  White Tophat (Head): $15.33
    #Torso  -  Lab Coat (Torso): $10.22
    #Legs  -  Wool Leggings (Legs): $16.37
    #Feet  -  Rubber Boots (Feet): $20.91

    print()
    outfit2 = cheap_outfit([Store('Blacksmith', 'blacksmith.csv'),
Store('Sparkles', 'sparkles.csv')])
    for key in outfit2:
        print(key,' - ',outfit2[key])
    #Head  -  Bedazzled Monocle (Head): $11.37
    #Torso  -  Bronze Cuirass (Torso): $24.13
    #Legs  -  Steel Battle-Skirt (Legs): $11.72
    #Feet  -  Cyan Roller-Blades (Feet): $4.62

    print()
    outfit3 = cheap_outfit([Store('Sparkles', 'sparkles.csv'),
Store('Goth City', 'gothcity.csv'), Store('Blacksmith',
'blacksmith.csv'), Store('Professional Wear',
'professionalwear.csv')])
    for key in outfit3:
        print(key,' - ',outfit3[key])
    #Head  -  Batman Mask (Head): $6.56
    #Torso  -  Lab Coat (Torso): $10.22
    #Legs  -  Steel Battle-Skirt (Legs): $11.72
    #Feet  -  Cyan Roller-Blades (Feet): $4.62
```