Name: _____

Problem	Points	Score
1	50	
2	50	
Total	100	

Notes:

(1) Please see the instructions sent in email to the class for how to submit your work.

(2) You are allowed to use all the web resources at your disposal except other human beings (and talking to someone via a chat line counts as an interaction with a human being ☺

(3) In addition to providing your code, explain your solution to each problem in the comments in your code, especially if you want partial credit.

(4) YOU MUST PROGRAM IN PYTHON!

(5) You must rsync your solutions to electrodata using the subdirectory name "ex_03."

Problem No. 1: Create a directory named p01. In this directory, create a file foo_00.py that contains a class definition and implementation called Vehicle that has the following attributes:

- data:
  - manufacturer: the name of the manufacturer (e.g., "Ford")
  - model: the name of the model (e.g., "Focus")
  - width: the width of the wheel base in inches
  - length: the length of the car in inches
- public methods include:
  - constructor with no arguments: sets all internal data to "-1"
  - constructor with arguments: for all the protected data
  - debug: prints out the values of all internal data in the class.

Also create a main program called foo.py that supports the following interface:

```
python foo.py foo.txt
```

where foo.txt contains information in this format:

```
Vehicle = Ford, Focus, 78.0, 115.0
Vehicle = Chevrolet, Volt, 65.0, 88.0
Vehicle = Mazda, Rx8, 55.0, 72.0
```

You should be able to specify an arbitrary number of entries.

Your main program should read the entries from the file and store them in a list. It should then loop over all entries in the list and call the debug method for each vehicle. Your output should print the data in this format:

```
Vehicle:
  manufacturer = …
  model = …
  width = …
  length = …
Vehicle:
  manufacturer = …
  model = …
  width = …
  length = …
```

The loop to read data from the file should be separate from your debug loop.

Problem No. 2: Create a directory p02 (on electrodata, the path is going to be ~picone/rje/ece_3822/lastname_firstname/ex_03/p02). Create a function in a file, foo.py, in this directory, that remembers all of the values that have been used previously in calling it, and prints out the sum of these values. For example, if I call this function from the command line in this sequence of steps:

```
cd ~picone/rje/ece_3822/lastname_firstname/ex_03/p02
python foo.py 1          => produces an output of "sum = 1"
python foo.py 2 3        => produces an output of "sum = 6"
python foo.py 3.5 0 9    => produces an output of "sum = 18.5"
```

Your program must obviously work for any values. This is the easy part…

There is a twist, however ☺ I must be able to call your program from any location:

```
cd ~picone;
python ~picone/rje/ece_3822/picone_joseph/foo.py 1      => produces an output of "sum = 1"
python ~picone/rje/ece_3822/picone_joseph/foo.py 2      => produces an output of "sum = 3"
python ~picone/rje/ece_3822/picone_joseph/foo.py 3 4    => produces an output of "sum =10"
```

INCLUDING a directory for which I don't have write permission (such as a system directory). For example, if I cd to /usr, the above commands should still work.

There is yet another twist ☺ The following sequence of commands should work:

```
cd ~picone;
python ~picone/rje/ece_3822/picone_joseph/foo.py 1 => produces an output of "sum = 1"
cd /data/isip:
python ~picone/rje/ece_3822/picone_joseph/foo.py 2 => produces an output of "sum = 3"
```

Note that I am changing directories between calls. That is an **IMPORTANT** detail.