COMP 348: ASSIGNMENT 2

DUE DATE: 11:55 PM, Sunday, March 18

Note that assignments must be submitted on time in order to received full value. Appropriate late penalties (< 1 hour = 10%, < 24 hours = 25%) will be applied, as appropriate.

DESCRIPTION: In this assignment, you will have a chance to try your hand with the Python programming language. Your job will be to develop a small database server. Ordinarily, this would be a significant undertaking. However, with Python, this will be surprisingly straightforward and requires a relatively modest amount of code. Your DB system will work as follows:

1. You will construct a client/server application. In your case, only one client program will access the DB, so you do not have to worry about issues like concurrency or thread control. It's just a one-to-one form of communication. Python provides a SocketServer class in its standard libraries. You can use this to get started, along with the sample code provided in the Python docs (you can use the 9999 port for the DB server).

2. In addition to listening for client requests, the server program must first load the database and provide access to the data. The data base will be loaded from a simple, plain text disk file called `data.txt`. In your case, the database will hold customer records. A customer record will be a tuple with the following format

   name, age, address, phone#

   To record this information on disk, you will store one record per line, and separate each field with a bar symbol ('|'). A simple 3 record database might look like this

   ```
   John|43|123 Apple street|514 428-3452
   Katya|   26|49 Queen Mary Road|514 234-7654
   Ahmad|91|1888 Pepper Lane|
   ```

   Note that these values are very simple (first name, street address only, etc.). Also note that some basic checking must be done when values are read. For example, additional leading or trailing spaces may be present (e.g., Katya's age field), and some fields may not have a value (e.g., Ahmad's phone#). This is okay, as long as the name field is provided. If the name is missing, the record should be skipped. In any case, you should provide some basic error checking to make sure that your code doesn't simply crash in these cases.

   For the assignment, you should create a text file with about 20 records. The marker will use a data set of the same general format for grading purposes.

3. When the server is started, the first thing it will do is load the data set into memory (The database file should be stored in the same folder as the server application). Each tuple will be stored in an in-memory data structure. Feel free to use any standard Python data structure or combination of data structures for this purpose. In your case, you must lookup customers by name (these are case sensitive matches). So if I want to see the information for customer "John", for example, this should be a very simple operation.

4. You will also build a simple client application to access the server. Again, you can use the sample code on the python website as a starting point (you will likely want to use the `socket` object to connect to the `socketserver`). The idea is that you will send a request (just a text string) to the server and ask for a specific task to be performed.

The use interface (provided by the client) will consist of the following menu:

```
Python DB Menu

1. Find customer
2. Add customer
3. Delete customer
4. Update customer age
5. Update customer address
6. Update customer phone
7. Print report
8. Exit

Select:
```

This may look like a lot of options, but each of these is quite trivial. The `Select` prompt at the bottom will wait for a number to be entered. When a valid menu option is provided, the client application will then take the appropriate action as follows:

1. Find customer: Prompt the user for a name and then send the name to the server. The server will respond to the client either with the full customer record or a "customer not found" message. Note that this message MUST be sent back to the client which will then display it. The server itself should never print anything directly since, in theory, it could be placed on another machine. The menu will then be displayed again so that another choice can be made.

   For example, if the user enters the name "John", you would either see something like:

   ```
   Customer Name: John
   Server response: John|34|ABC Street|768-3245
   or
   Server response: John not found in database
   ```

2. Add customer: The client will prompt the user to enter each of the four fields. These will be sent to the server and the server will respond with either a “Customer already exists” message or a confirmation message that the customer has been added.

3. Delete customer: Delete the specified customer. The server will respond with either a “Customer does not exist” message or a confirmation message.

4. Update options: all update options will prompt for a name, then for the field to be updated. The server will respond either with a “Customer not found” message or a confirmation message.

5. Print Report: This will print the contents of the database, sorted by name. Note that the sorted contents should be sent by the server back to the client and then displayed by the client app. The Print Report function is mandatory because it is the primary way for the grader (and you) to determine if the database server is working correctly. Specifically, you can display the database before you perform any updates, then check it again later to make sure everything is working. You can NOT get a passing grade on the assignment if the `Print Report` option does not work!

   So the report might look like this:

   ```
   ** Python DB contents **
   John|34|ABC Street|768-3245
   Bob|26|123 Doop Road|345-5678
   Donna|35|Here Road|564-6879
   Ahmad|43|67 Drury Lane|897-3456
   Nancy|21|Stuffy Street|345-7896
   Billy|199|123 Apple Street|435-456-5768
   sue|45|Happy Lane|456-3245
   ```

6. Exit does the obvious thing. It should print a “Good bye” message and simply terminate the client application. Note that the server process will still be running since it executes in an infinite loop. If the client is restarted, it should reconnect with the existing server. If you want to stop the server, you can do so either from the development GUI (e.g., Eclipse ‘kill’ button), or from the command line with a ‘kill’ signal (on Linux, this would be `kill processID`)

So that’s the basic idea. If you haven’t written Python code before, you will quickly see that it is a very accessible language with a lot of documentation and supporting materials online. If you like to code, this assignment should actually be fun, certainly less painful than the C assignment.

DELIVERABLES: Your submission will have multiple source files. At the very least it will have a file called `server.py` and `client.py`. Feel free to add additional python files to more easily manage your code. Do not include the `data.txt` file since the marker must use their own test set to ensure that all students are evaluated in the same way.

Once you are ready to submit, place the .py files into a zip file. The name of the zip file will consist of "a2" + last name + first name + student ID + ".zip", using the underscore character "_" as the separator. For example, if your name is John Smith and your ID is "123456", then your zip file would be combined into a file called a1_Smith_John_123456.zip". The final zip file will be submitted through the course web site on Moodle. You simply upload the file using the link on the assignment web page.

FINAL NOTE: While you may talk to other students about the assignment, all code must be written individually. Any sharing of assignment code will likely lead to an unpleasant outcome.

Good Luck