Important:

1] Assignments must be submitted on time in order to receive full value. Appropriate late penalties (< 1 hour = 10%, < 24 hours = 25%) will be applied.

2] It is the student's responsibility to verify that the assignment has been properly submitted. You can NOT send an alternate version of the assignment at a later date, with the claim that you must have submitted the wrong version at the deadline. Only the original submission will be graded (though you can certainly re-submit multiple times before the due date)

3] These are individual assignments. Feel free to talk to one another and even help others understand the basic ideas. But the source code that you write must be your own.

*4]*** This assignment will be written in Python. It is a very popular language and, as a result, there are enormous amounts of 3rd party Python code that can be downloaded from the web (e.g., PyPI – the Python Package Index). YOU CAN NOT USE ANY OF THAT. The purpose of the assignment is to help you become familiar with the basic form of the language – it is not for you to simply download libraries that someone else has written. So 100% of the code you write must work with the standard Python distribution (it already includes a huge amount of supporting code). The graders will NOT download any external Python packages in order to run your programs – if it doesn't run with the standard distribution, it will not be graded.*

*5] *** As previously noted, Python has two versions in common use: 2.x and 3.x. While they are quite similar, they are not completely compatible. For this assignment, make sure that you are using the more recent version, Python 3.x. This is what the graders will use for the evaluation.*

DESCRIPTION: In this assignment, you will have a chance to gain experience with the Python programming language. Python is an easy to use, dynamically typed Object Oriented language. It is syntactically similar to C-style languages like C++, C# and Java, but is somewhat simpler to understand, at least at the modest level of skill required for this assignment.

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 trivial 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 (and various other network facilities as well). You can use this to get

started if you like, 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 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 at all (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. (We do not provide the testing set to students beforehand but there are no surprises in the test set – it is exactly as described above)

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. Python provides a dictionary structure for this purpose. In your case, you must look up customers by name. 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 trivial client application to access the server. Again, you can use the sample code on the main python website as a starting point. 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 possible requests are as follows:

   ```
   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. To make it easier to send commands, you will present a menu type interface. 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.

2. Add customer: The client will prompt the user to enter each of the four fields. These will be collected and 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 <u>mandatory</u> 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!

   As noted, no 3rd party libraries can be used. So do not download and install some sort of Python reporting/table function here. You must write the reporting code yourself and provide basic formatting to ensure that the output is readable.

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.

So that's the basic idea. If you haven't used Python before, you will find 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.

DELIVERABLES: Your submission will have exactly two source files. Specifically, it will have a file called `server.py` and a filed called `client.py`. Do not include the `data.txt` file since the marker must use his/her own test set to ensure that all students are evaluated in the same way.

Do NOT use any additional packages or folders. Doing so can be very problematic for the graders since they need to be able to run a huge number of programs, without having to deal with unexpected path/navigation errors. So make sure that your code will simply run from the current folder.

Once you are ready to submit, combine your two source files into a zip file. The name of the zip file will consist of "a1" + 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: As I have mentioned before, you are free to develop the Python code however you like: by downloading and setting up a traditional development environment on your own machine (it's not hard at all) or by using an online system like `repl.it`. If you find that the simple online approach works for you, then you need not do any more than that, as long as you meet the requirements of the assignment.

Good Luck