**Part A) Write a web service**

hints:

* if you're using python, look at wsgiref or flask
* if you’re using java, look at Jetty or grizzly and Jersey

Using a language of your choice, create a web service which:

* is a key/value store (don't worry about being fancy with the storage mechanism; it could just be in memory)
* uses RESTful methods to get/set/update/delete values

Implement the following methods in web service:

1) http POST to <http://localhost/kvstore>

* this method is used to create a new key/value pair stored in the web service
* the POST body should be a json object with the appropriate content-type of the form { "key": "myname", "value": "myvalue" } where myname and myvalue are the name of the key and the value of the key respectfully
* the method should return http codes 409 Conflict if the object already exists and 201 Created if creation was successful, and the body of the response should be json of the form described above.

2) http PUT to <http://localhost/kvstore>

* this method is used to update already-existing key/value pairs stored in the web service
* the PUT body should be a json object with the appropriate content-type of the form { "key": "myname", "value": "myvalue" } where myname and myvalue are the name of the key and the value of the key respectfully
* if the key/value pair does not exist, this method should return 404 (not found)
* if the key/value pair exists, this method should update the value and return 200 OK, and the body of the response should be json of the form specified above.

3) http GET to [http://localhost/kvstore/<myname>](http://localhost/kvstore/%3Cmyname%3E)

* this method is used to fetch the value of a key named "myname"
* if the key/value pair exists, the name and value should be returned as described above, and the http code should be 200 OK
* the body of the response should be a json string of the form { "key": "myname", "value": "myvalue" }
* if the key/value pair does not exist, the method should return 404 Not Found

4) http GET to <http://localhost/kvstore>

* this method is used to fetch all of the key/value pairs
* if the key/value pair exists, the name and value should be returned as described above, and the http code should be 200 OK
* the body of the response should be a json string of the form [{ "key": "myname1", "value": "myvalue1" }, { "key": "myname2", "value": "myvalue2" }]
* if there are no key/value pairs, then 200 OK should be returned with an empty json array

5) http DELETE to [http://localhost/kvstore/<myname>](http://localhost/kvstore/%3Cmyname%3E)

* this method is used to delete a key of the name "myname" and its value
* if the key exists, the key should be deleted from the key/value store, the name and value of the deleted key should be returned as described above, and the http code should be 200 OK
* the body of the response should be the key and value that were deleted, of the form { "key": "myname", "value": "myvalue" } where the key name is "myname" and the value is "myvalue"
* if the key does not exist, the method should return an http code of 404 Not Found

II) Using a language of your choice, implement a test harness that:

* starts the web service
* tests doing unexpected http operations does not have any side-effect
* stops the web service

**Part B) Writing automated tests**

In automated test scripts, we will look for following:

1. Thinking of tests that need to be automated
2. Construction and flow of automated tests
3. Code re-usability.
4. Please add comments wherever required to help us understand what each test is doing.
5. If requesting Web service script, please write automated tests based on the data in the web service.

**Optional tasks (Bonus points)**

1. API calls are made concurrently from multiple clients.
2. Any HTTP operation other than GET, POST, PUT and DELETE should give 405.
3. What other tests you can think of? Please write (no coding required).

**Finally, please provide comprehensive instruction document to run the entire test harness, which includes dependencies etc.**