/

INSTRUCTIONS FOR EXECUTION

1. For basic client-server interaction:
   There are two python scripts named client.py and server.py.
   Client.py: It takes a request as input and sends it to the server. After the server's response, it displays the response of the server.
   Server.py: It accepts request from client, parses it, and performs the GET/PUT/DELETE tasks accordingly and sends back response to client.A dictionary is declared to store the data.
   To execute:
   A. Firstly, open the terminal and go to location home/tutorials/exercises/basic and run commands "make clean" and "make run".
   B. Now, we will be in mininet environment, run 'xterm h1' and 'xterm h2' to open terminals of host1 and host2. Now, h1 is the client and h2 is the server.
   C. First run server.py in h2 terminal. For server IP, enter "10.0.1.2" as input. A connection gets opened.
   D. Run client.py in h1 terminal. Provide server IP as "10.0.1.2". Now, provide the request as input.
   Input types:a. **PUT:**
   Input: PUT,KEY,VALUE
   Request format: **"PUT /assignment1/key/val HTTP/1.1"**
   On execution,output format: **"HTTP/1.1   key:valSTORED 200 OK"**
   B. **GET:**
   Input: GET,KEY
   Input format: **"GET /assignment1?request=key HTTP/1.1"**
   Output format:**"HTTP/1.1 val 200 OK"**
   C. **DELETE:**
   Input: DELETE,KEY
   Input format:**"DELETE /assignment1/key HTTP/1.1"**
   Output format: **"HTTP/1.1 DELETED 200 OK"**
   D. **On other invalid inputs:**
   Output : **"HTTP/1.1 400 Bad Request"**

2. Web cache development:

There are three python scripts named client.py, cache.py and server.py.

Client.py: It takes input and sends it to cache. After the cache's response, it prints the output.

Cache.py: Accepts request from client. If it is PUT, it sends a request to the server. If the request is DELETE, it deletes the key in cache's memory and sends the same request to the server. If it is GET, it checks in the cache memory, if present , returns to the client, else sends the same request to the server.A dictionary is declared to store the data.

Server.py: Accepts requests from cache and responds in the same way as mentioned in above case.

To execute:

A. Follow the same commands as above at home/tutorials/exercises/star, but this time, run xterm three times for h1,h2 and h3.
B. First open the connection in h3(server) with IP address input as "10.0.1.3".Now, run cache.py, provide cache ID as "10.0.1.2" and server ID as "10.0.1.3".
C. Run the client.py in h1 terminal(server IP is of cache i.e., 10.0.1.2) and provide a request as input.Request types:
    a. PUT:Input:PUT,KEY,VALUE
    "PUT /assignment1/key/val HTTP/1.1"
    Output : **"HTTP/1.1 PUT key:val 200 OK"**
    b. GET:Input:GET,KEY
    "GET /assignment1?request=key HTTP/1.1"
    Output: **"HTTP/1.1 GET key val 200 OK"**
    If key is not present in both cache and server, output will be
    "HTTP/1.1 Value not found 400 Bad Request"
    c. DELETE:Input:DELETE,KEY
    "DELETE /assignment1/key HTTP/1.1"
    Output: **"HTTP/1.1 DELETE 200 OK"**

In this case, every request from client first goes to cache and then to server if required.