☆ Pseudo code of sample applications

For learning purposes, we encourage you to build the following four applications. The setup for these are shown in the **Useful-Mininet-Setups.pdf**

1. Hub
Description: A hub will flood any packet that arrives on a particular port of a switch out of all the other ports.

Packet Logic:

- On *packet_in* to a switch,
  - Make list of all ports on the switch
  - Send packet out of all those ports, except the in_port

2. MAC Learning Switch
Description: The learning switch keeps track of where the host with each MAC address is located and accordingly sends packets towards the destination and not FLOOD it like a hub.

Packet logic:

- Create a dictionary (or HashMap in Java) called mac_to_port
  - For multiple switches in the network, you need a dictionary per switch
- On *packet_in* to a switch s1,
  - Parse packet to reveal src and dst MAC addr
  - Store in the dictionary the mapping between src_mac and the in_port
  - Lookup dst_mac in mac_to_port dict of switch s1 to find next hop
  - If next hop is found, create flow_mod and send
  - Else, flood like hub.

3. Stateless Load-balancer
Description: Server load-balancer that maps incoming HTTP requests to a different server in a pre-determined fashion. Main aspect is that the clients only know the IP address of the load-balancer and not the real servers.

Packet logic:

- Pick static virtual_ip (10.0.0.5), virtual_mac (00:00:00:00:00:05) of load-balancer
- Initialize list of servers and their MAC.
- On *packet_in* for virtual_ip from client "X",
  - Pick server "Y" in round-robin fashion
  - Insert flow:
    - Match: Same as the incoming packet
    - Actions: 1) Rewrite dst_mac, dst_ip of packet to that of "X", 2) Forward to Mac_to_port["X"]
  - Proactively insert reverse flow:
    - Match: Src (IP, MAC, TCP_Port) = Y, Dst = X,
    - Action: 1) Rewrite src_mac, src_ip to that of virtual_ip, 2) Forward to port towards "X"
  - All subsequent packets of the request will directly be sent to the chosen server and not be seen by the controller.

4. Content-aware Load-balancer
Description: Based on the exact web page requested, this server load-balancer maps incoming HTTP requests to a different server in a pre-determined fashion.

Packet logic: Compared to the above stateless load-balancer, this load-balancer needs to select the server "Y" and push the bi-directional rules only after the first "GET" request packet arrives on the switch. This means that the first 3 *packet_in* from the TCP handshake (SYN, SYNC ACK, ACK) will need to be handled using *packet_out* commands by the controller.