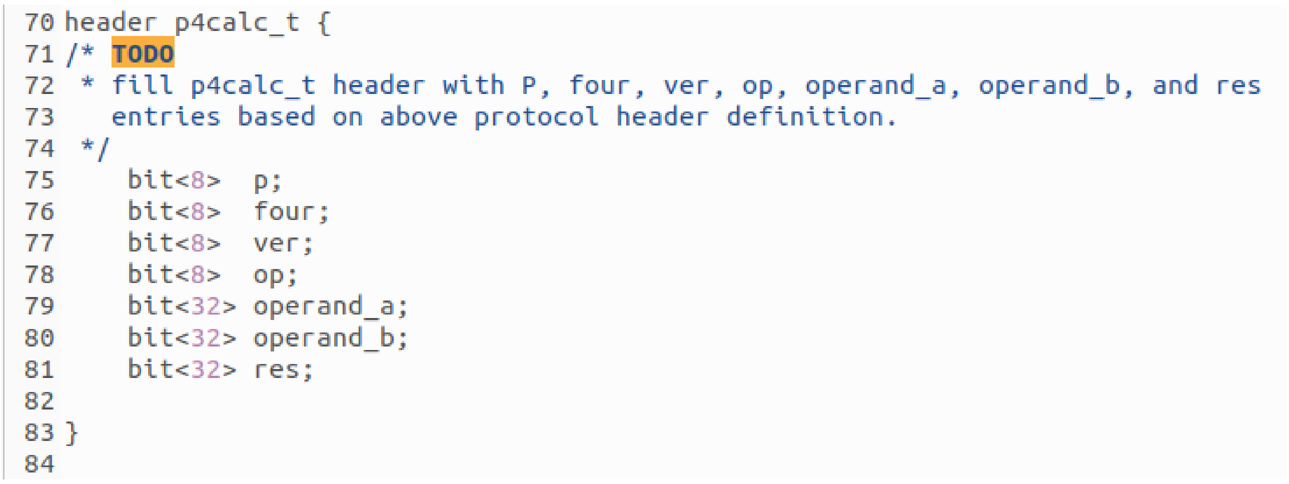
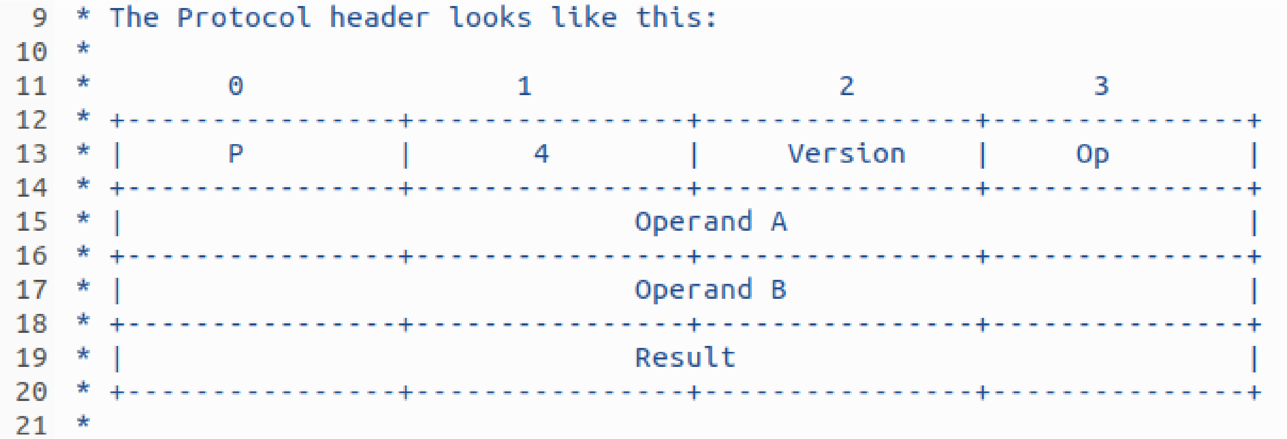
Exercise 5)

Changes to calc.p4:



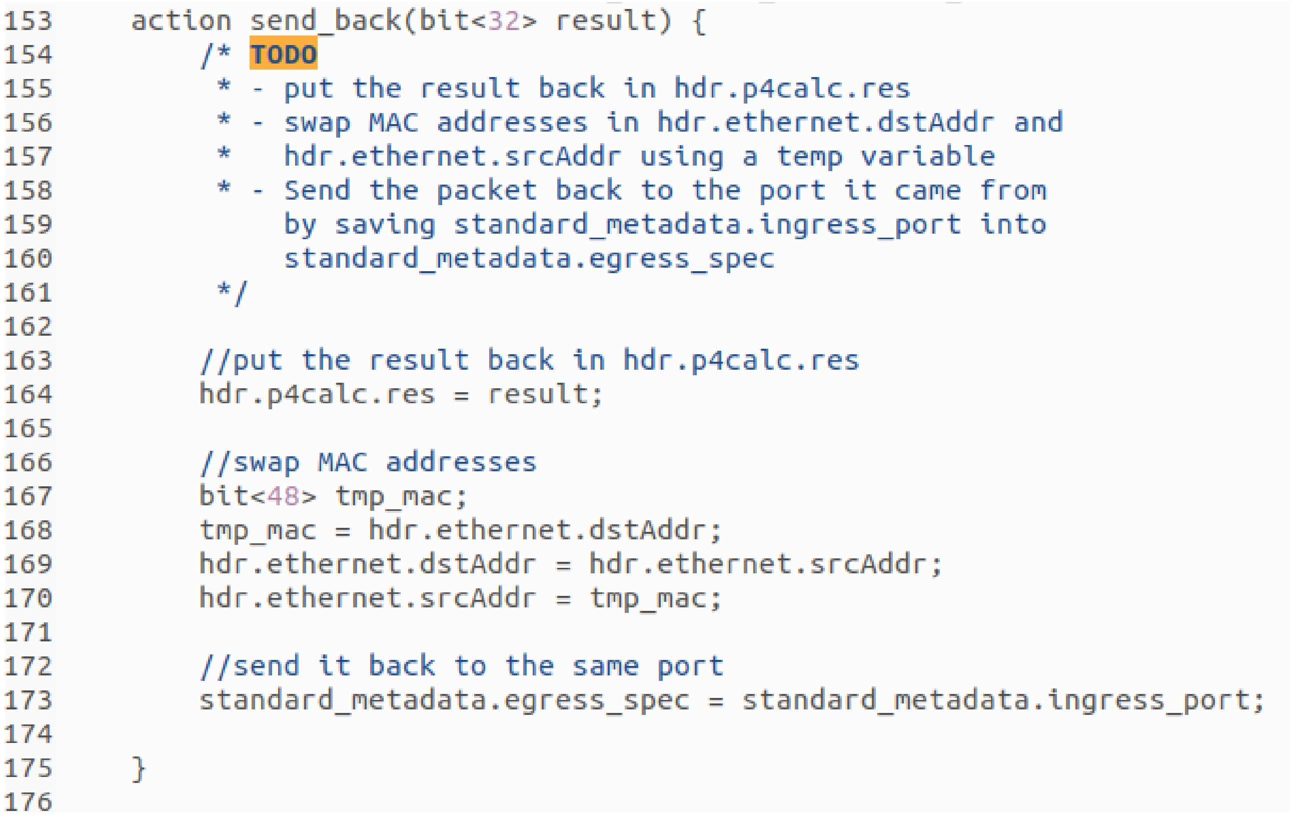
```
header p4calc_t {
/* TODO
 * fill p4calc_t header with P, four, ver, op, operand_a, operand_b, and res
   entries based on above protocol header definition.
 */
    bit<8>  p;
    bit<8>  four;
    bit<8>  ver;
    bit<8>  op;
    bit<32> operand_a;
    bit<32> operand_b;
    bit<32> res;

}

```

I filled in the header with these bit sizes because p, four, ver and op are 2 digit hexadecimal numbers so they are 8 bits.



```
 * The Protocol header looks like this:
 *
 *        0                1                  2              3
 * +----------------+----------------+----------------+---------------+
 * |      P         |       4        |     Version    |     Op        |
 * +----------------+----------------+----------------+---------------+
 * |                              Operand A                           |
 * +----------------+----------------+----------------+---------------+
 * |                              Operand B                           |
 * +----------------+----------------+----------------+---------------+
 * |                              Result                              |
 * +----------------+----------------+----------------+---------------+
 *
```

operand_a, operand_b and result are 4 times the length of p, four, ver and op so they must be 32 bits each.

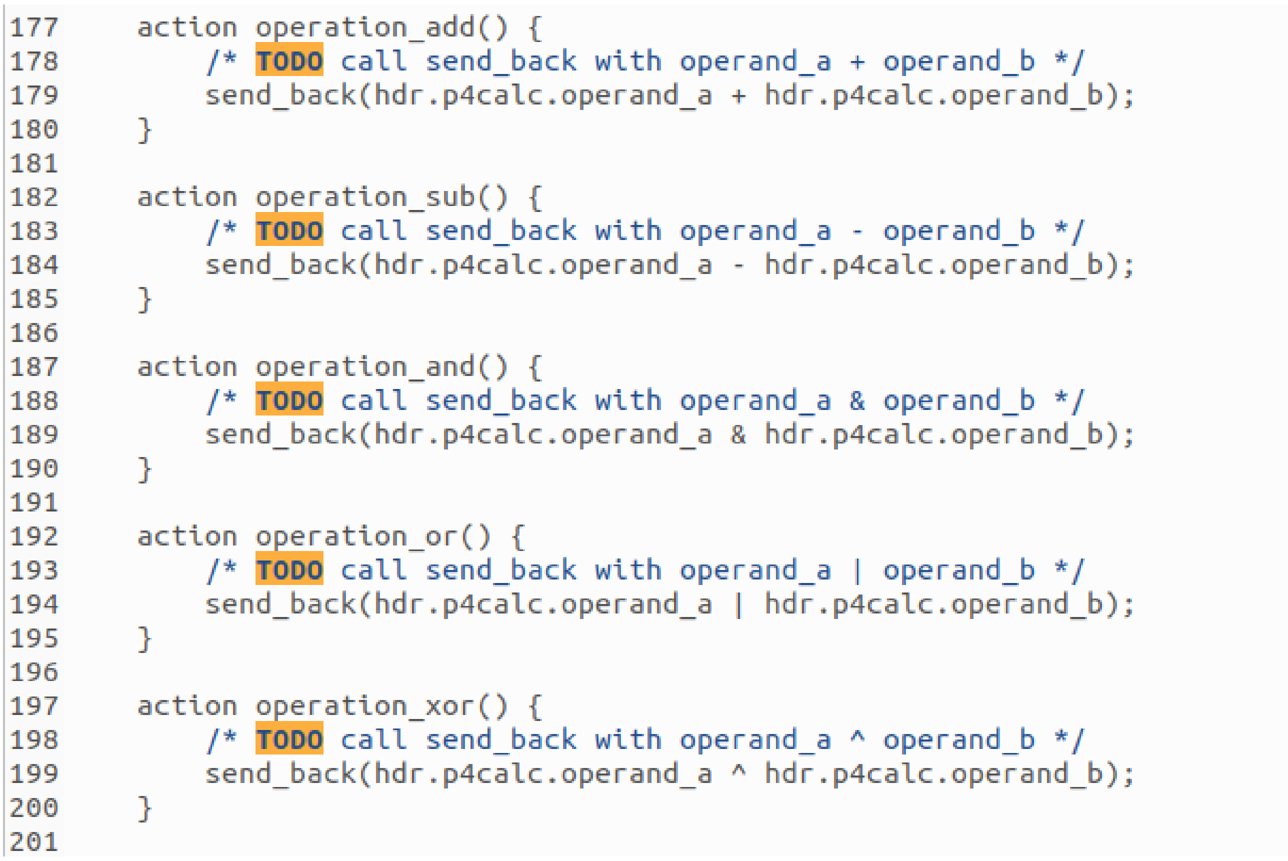


```
    action send_back(bit<32> result) {
        /* TODO
         * - put the result back in hdr.p4calc.res
         * - swap MAC addresses in hdr.ethernet.dstAddr and
         *   hdr.ethernet.srcAddr using a temp variable
         * - Send the packet back to the port it came from
             by saving standard_metadata.ingress_port into
             standard_metadata.egress_spec
         */

        //put the result back in hdr.p4calc.res
        hdr.p4calc.res = result;

        //swap MAC addresses
        bit<48> tmp_mac;
        tmp_mac = hdr.ethernet.dstAddr;
        hdr.ethernet.dstAddr = hdr.ethernet.srcAddr;
        hdr.ethernet.srcAddr = tmp_mac;

        //send it back to the same port
        standard_metadata.egress_spec = standard_metadata.ingress_port;

    }

```

I stored result into the res variable we made earlier by using its location hdr.p4calc.res

I swapped the mac address by first defining a temporary variable with the same length as a mac address (6 2-digit hexadecimal numbers so 6 x (2 x 4) so 48 bits).

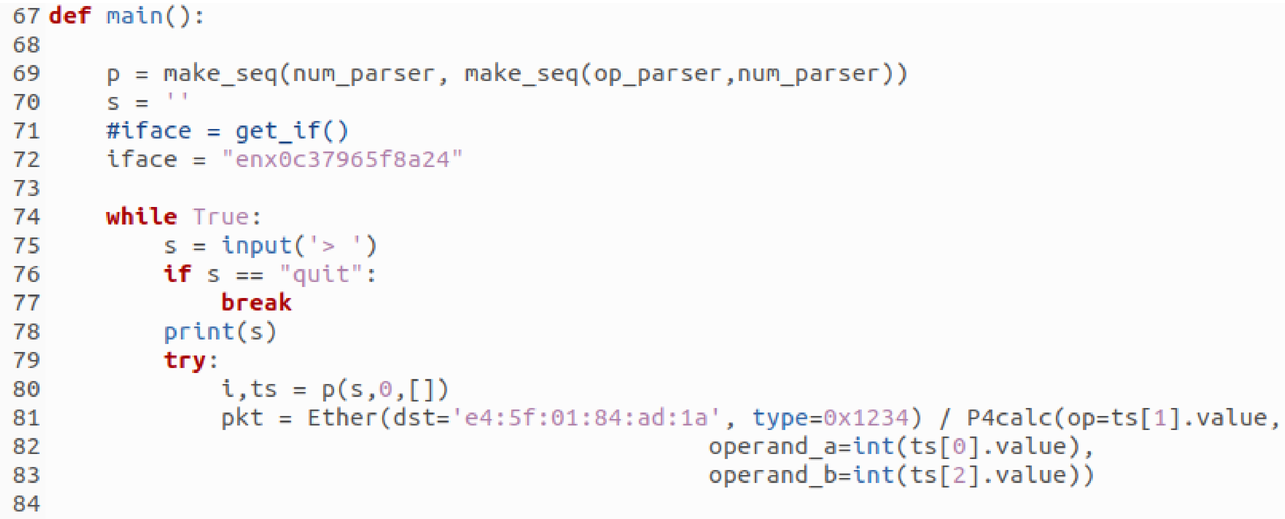
I sent the packet back to the same port as instructed.



```
    action operation_add() {
        /* TODO call send_back with operand_a + operand_b */
        send_back(hdr.p4calc.operand_a + hdr.p4calc.operand_b);
    }

    action operation_sub() {
        /* TODO call send_back with operand_a - operand_b */
        send_back(hdr.p4calc.operand_a - hdr.p4calc.operand_b);
    }

    action operation_and() {
        /* TODO call send_back with operand_a & operand_b */
        send_back(hdr.p4calc.operand_a & hdr.p4calc.operand_b);
    }

    action operation_or() {
        /* TODO call send_back with operand_a | operand_b */
        send_back(hdr.p4calc.operand_a | hdr.p4calc.operand_b);
    }

    action operation_xor() {
        /* TODO call send_back with operand_a ^ operand_b */
        send_back(hdr.p4calc.operand_a ^ hdr.p4calc.operand_b);
    }

```

I used the send_back action by replacing the parameter with the appropriate operation of operand_a and operand_b.

Changes to calc.p4:

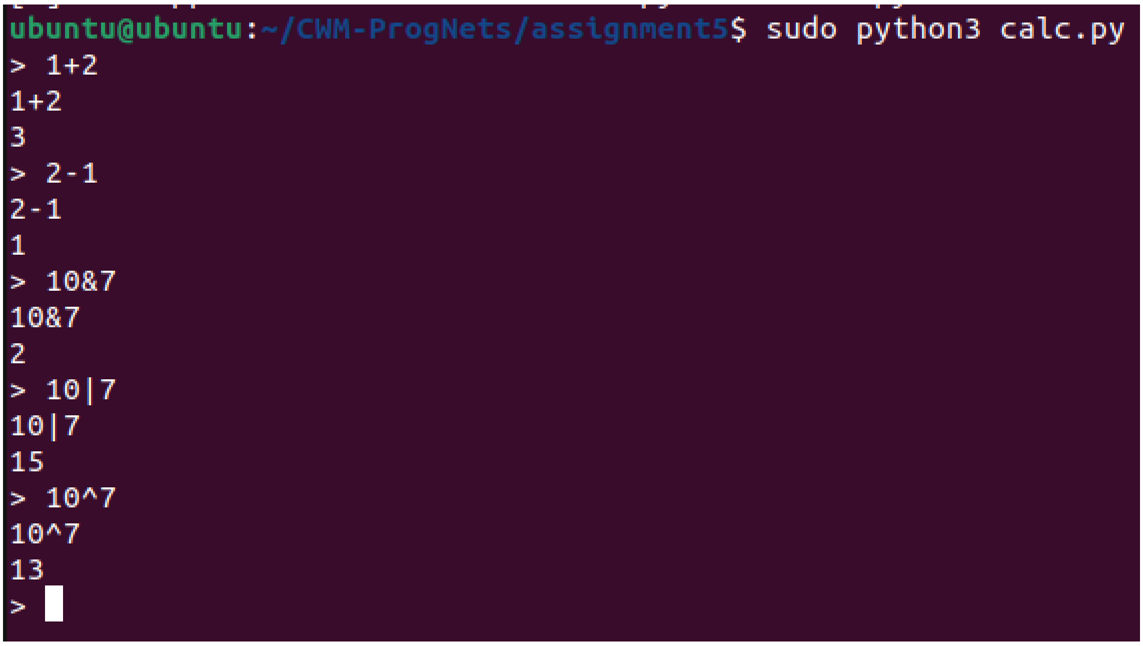


```
def main():

    p = make_seq(num_parser, make_seq(op_parser,num_parser))
    s = ''
    #iface = get_if()
    iface = "enx0c37965f8a24"

    while True:
        s = input('> ')
        if s == "quit":
            break
        print(s)
        try:
            i,ts = p(s,0,[])
            pkt = Ether(dst='e4:5f:01:84:ad:1a', type=0x1234) / P4calc(op=ts[1].value,
                                              operand_a=int(ts[0].value),
                                              operand_b=int(ts[2].value))

```

I changed the interface name to the interface of the lab machine because it runs calc.py

I changed the destination mac address of the packet to the mac address of the Raspberry PI because the lab machine will send packets to the Raspberry PI so it can get the relevant packets back from calc.p4

Example inputs:



```
ubuntu@ubuntu:~/CWM-ProgNets/assignment5$ sudo python3 calc.py
> 1+2
1+2
3
> 2-1
2-1
1
> 10&7
10&7
2
> 10|7
10|7
15
> 10^7
10^7
13
> 
```

All of the outputs are as expected.

Link to directory:
https://github.com/VKing15/CWM-ProgNets.git