Task 1: Develop a program that capture 10 Ethernet frames from the network and display the source and destination MAC addresses along with the data in each of the frames.

Program:

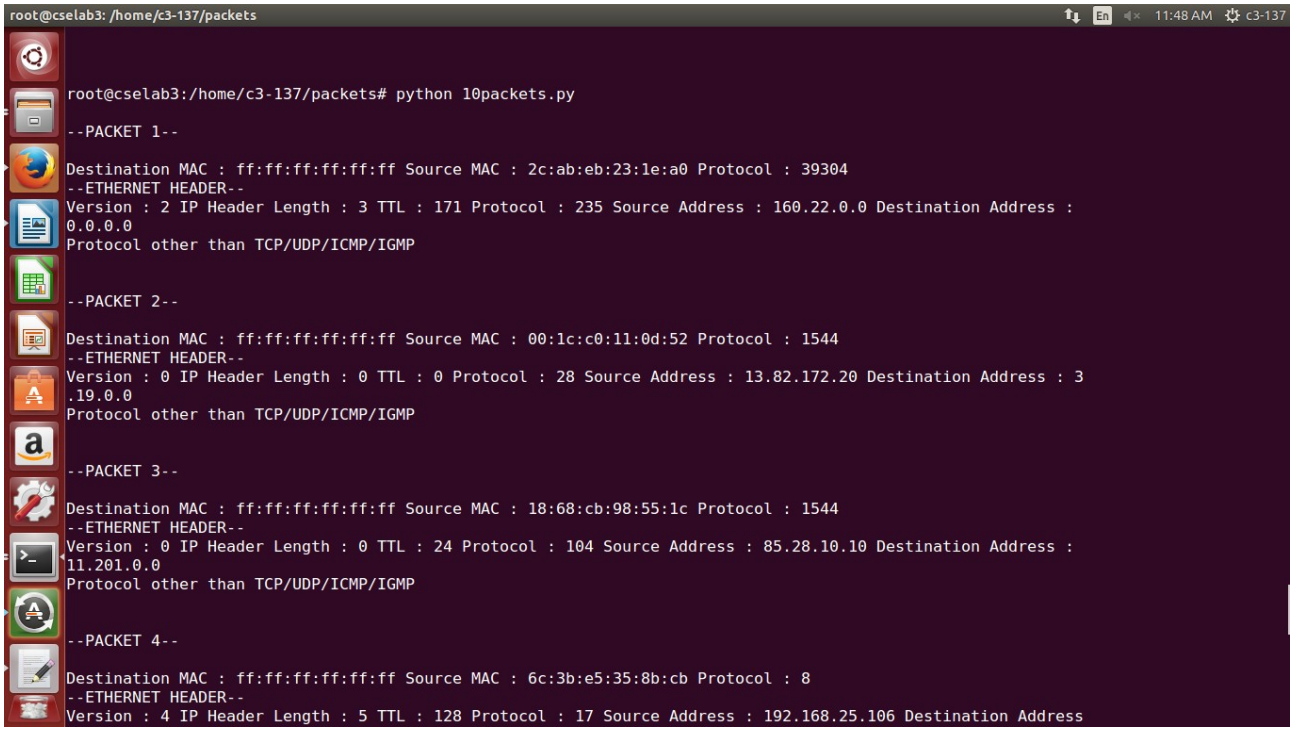
```
import socket, sys
from struct import *

#Convert a string of 6 characters of ethernet address into a dash
separated hex string
def eth_addr (a) :
     b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) ,
ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))
     return b

#create a AF_PACKET type raw socket (thats basically packet level)
#define ETH_P_ALL   0x0003          /* Every packet (be
careful!!!) */
try:
     s = socket.socket( socket.AF_PACKET , socket.SOCK_RAW ,
socket.ntohs(0x0003))
except socket.error , msg:
     print 'Socket could not be created. Error Code : ' +
str(msg[0]) + ' Message ' + msg[1]
     sys.exit()
# receive a packet
for i in range(0, 10):
     print
     print '--PACKET '+str(i+1)+'--'
     print
     packet = s.recvfrom(65565)

     #packet string from tuple
     packet = packet[0]

     #parse ethernet header
     eth_length = 14

     eth_header = packet[:eth_length]
     eth = unpack('!6s6sH' , eth_header)
     eth_protocol = socket.ntohs(eth[2])
     print 'Destination MAC : ' + eth_addr(packet[0:6]) + ' Source
MAC : ' + eth_addr(packet[6:12]) + ' Protocol : ' +
str(eth_protocol)

     print '--ETHERNET HEADER--'
     #Parse IP header
     #take first 20 characters for the ip header
     ip_header = packet[eth_length:20+eth_length]

     #now unpack them
     iph = unpack('!BBHHHBBH4s4s' , ip_header)
```

```
     version_ihl = iph[0]
     version = version_ihl >> 4
     ihl = version_ihl & 0xF

     iph_length = ihl * 4

     ttl = iph[5]
     protocol = iph[6]
     s_addr = socket.inet_ntoa(iph[8]);
     d_addr = socket.inet_ntoa(iph[9]);

     print 'Version : ' + str(version) + ' IP Header Length : ' +
str(ihl) + ' TTL : ' + str(ttl) + ' Protocol : ' + str(protocol) +
' Source Address : ' + str(s_addr) + ' Destination Address : ' +
str(d_addr)

     #TCP protocol
     if protocol == 6 :
          print '--TCP--'
          t = iph_length + eth_length
          tcp_header = packet[t:t+20]

          #now unpack them :)
          tcph = unpack('!HHLLBBHHH' , tcp_header)

          source_port = tcph[0]
          dest_port = tcph[1]
          sequence = tcph[2]
          acknowledgement = tcph[3]
          doff_reserved = tcph[4]
          tcph_length = doff_reserved >> 4

          print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Sequence Number : ' + str(sequence) + '
Acknowledgement : ' + str(acknowledgement) + ' TCP header length :
' + str(tcph_length)

          h_size = eth_length + iph_length + tcph_length * 4
          data_size = len(packet) - h_size

          #get data from the packet
          data = packet[h_size:]

          print 'Data : ' + data

     #ICMP Packets
     elif protocol == 1 :
          print '--ICMP--'
          u = iph_length + eth_length
          icmph_length = 4
          icmp_header = packet[u:u+4]

          #now unpack them :)
```

```
        icmph = unpack('!BBH' , icmp_header)

        icmp_type = icmph[0]
        code = icmph[1]
        checksum = icmph[2]

        print 'Type : ' + str(icmp_type) + ' Code : ' + str(code)
+ ' Checksum : ' + str(checksum)

        h_size = eth_length + iph_length + icmph_length
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data

    #UDP packets
    elif protocol == 17 :
        print '--UDP--'
        u = iph_length + eth_length
        udph_length = 8
        udp_header = packet[u:u+8]

        #now unpack them :)
        udph = unpack('!HHHH' , udp_header)

        source_port = udph[0]
        dest_port = udph[1]
        length = udph[2]
        checksum = udph[3]

        print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Length : ' + str(length) + ' Checksum : ' +
str(checksum)

        h_size = eth_length + iph_length + udph_length
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data

    #IGMP packets
    elif protocol == 88 :
        print '--IGMP--'
        u = iph_length + eth_length
        igmph_length = 8
        igmp_header = packet[u:u+8]

        #now unpack them :)
        igmph = unpack('!HHHH' , igmp_header)
```

```
        source_port = igmph[0]
        dest_port = igmph[1]
        length = igmph[2]
        checksum = igmph[3]

        print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Length : ' + str(length) + ' Checksum : ' +
str(checksum)

        h_size = eth_length + iph_length + igmph_length
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data

    #some other IP packet
    else :
        print 'Protocol other than TCP/UDP/ICMP/IGMP'

    print
```

Output



```
root@cselab3: /home/c3-137/packets                                11:49 AM  c3-137
--PACKET 4--

Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 6c:3b:e5:35:8b:cb Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 192.168.25.106 Destination Address
 : 192.168.25.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 4479
Data : ××□□□□FHFAEBEECACACACACACACACACACAAA □□


--PACKET 5--

Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 2c:ab:eb:22:4d:c4 Protocol : 39304
--ETHERNET HEADER--
Version : 2 IP Header Length : 3 TTL : 171 Protocol : 235 Source Address : 196.8.0.0 Destination Address : 0
.0.0.0
Protocol other than TCP/UDP/ICMP/IGMP


--PACKET 6--

Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 00:e0:4c:15:03:0f Protocol : 1544
--ETHERNET HEADER--
Version : 0 IP Header Length : 0 TTL : 0 Protocol : 224 Source Address : 3.15.172.20 Destination Address : 0
.1.0.0
Protocol other than TCP/UDP/ICMP/IGMP


--PACKET 7--

Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 2c:41:38:a8:ce:49 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 172.20.5.74 Destination Address :
255.255.255.255
--UDP--
Source Port : 55706 Dest Port : 3956 Length : 16 Checksum : 9037
```

```
root@cselab3: /home/c3-137/packets                                11:49 AM  c3-137
Source Port : 55706 Dest Port : 3956 Length : 16 Checksum : 9037
Data : B□□□□


--PACKET 8--

Destination MAC : 00:e0:4c:15:03:0f Source MAC : 34:64:a9:2b:68:88 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 64 Protocol : 1 Source Address : 172.20.1.181 Destination Address : 1
72.217.166.78
--ICMP--
Type : 8 Code : 0 Checksum : 6036
Data : □q××{Z×□□□□□□□□□□□□□□□□□□□□□!"#$%&'()*+,-./01234567


--PACKET 9--

Destination MAC : ff:ff:ff:ff:ff:ff Source MAC : 2c:41:38:ac:81:50 Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 128 Protocol : 17 Source Address : 172.20.3.200 Destination Address :
 172.20.7.255
--UDP--
Source Port : 137 Dest Port : 137 Length : 58 Checksum : 28683
Data : ×□□□□DBDJDCCACACACACACACACACACAAA □□


--PACKET 10--

Destination MAC : 34:64:a9:2b:68:88 Source MAC : 00:e0:4c:15:03:0f Protocol : 8
--ETHERNET HEADER--
Version : 4 IP Header Length : 5 TTL : 54 Protocol : 1 Source Address : 172.217.166.78 Destination Address :
 172.20.1.181
--ICMP--
Type : 0 Code : 0 Checksum : 8084
Data : □q××{Z×□□□□□□□□□□□□□□□□□□□□□!"#$%&'()*+,-./01234567

root@cselab3:/home/c3-137/packets#
```

Task 2: Develop a program that capture n packets and displays the complete packet details including the headers.

Program:

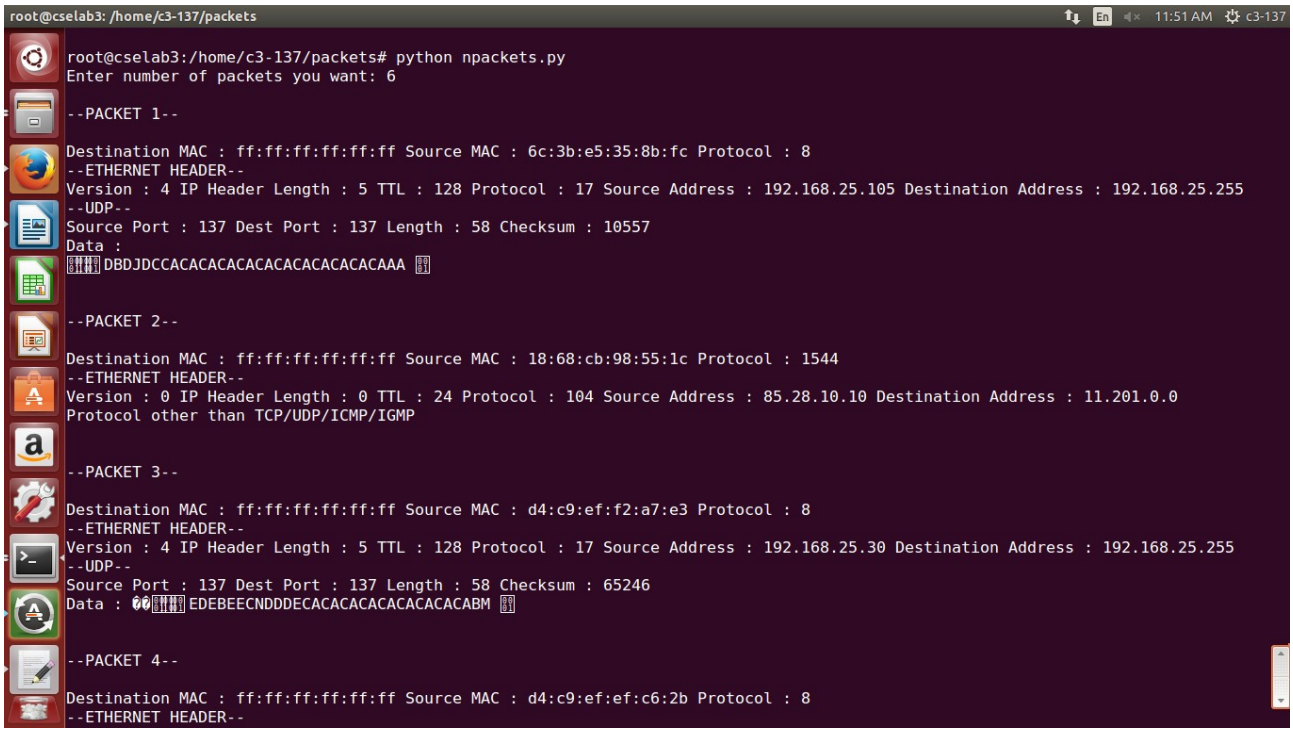
```
import socket, sys
from struct import *

#Convert a string of 6 characters of ethernet address into a dash
separated hex string
def eth_addr (a) :
     b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) ,
ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))
     return b

#create a AF_PACKET type raw socket (thats basically packet level)
#define ETH_P_ALL   0x0003          /* Every packet (be
careful!!!) */
try:
     s = socket.socket( socket.AF_PACKET , socket.SOCK_RAW ,
socket.ntohs(0x0003))
except socket.error , msg:
     print 'Socket could not be created. Error Code : ' +
str(msg[0]) + ' Message ' + msg[1]
     sys.exit()
n = input('Enter number of packets you want: ')
# receive a packet
for i in range(0, n):
     print
     print '--PACKET '+str(i+1)+'--'
     print
     packet = s.recvfrom(65565)

     #packet string from tuple
     packet = packet[0]

     #parse ethernet header
     eth_length = 14

     eth_header = packet[:eth_length]
     eth = unpack('!6s6sH' , eth_header)
     eth_protocol = socket.ntohs(eth[2])
     print 'Destination MAC : ' + eth_addr(packet[0:6]) + ' Source
MAC : ' + eth_addr(packet[6:12]) + ' Protocol : ' +
str(eth_protocol)


     print '--ETHERNET HEADER--'
     #Parse IP header
     #take first 20 characters for the ip header
     ip_header = packet[eth_length:20+eth_length]

     #now unpack them
```

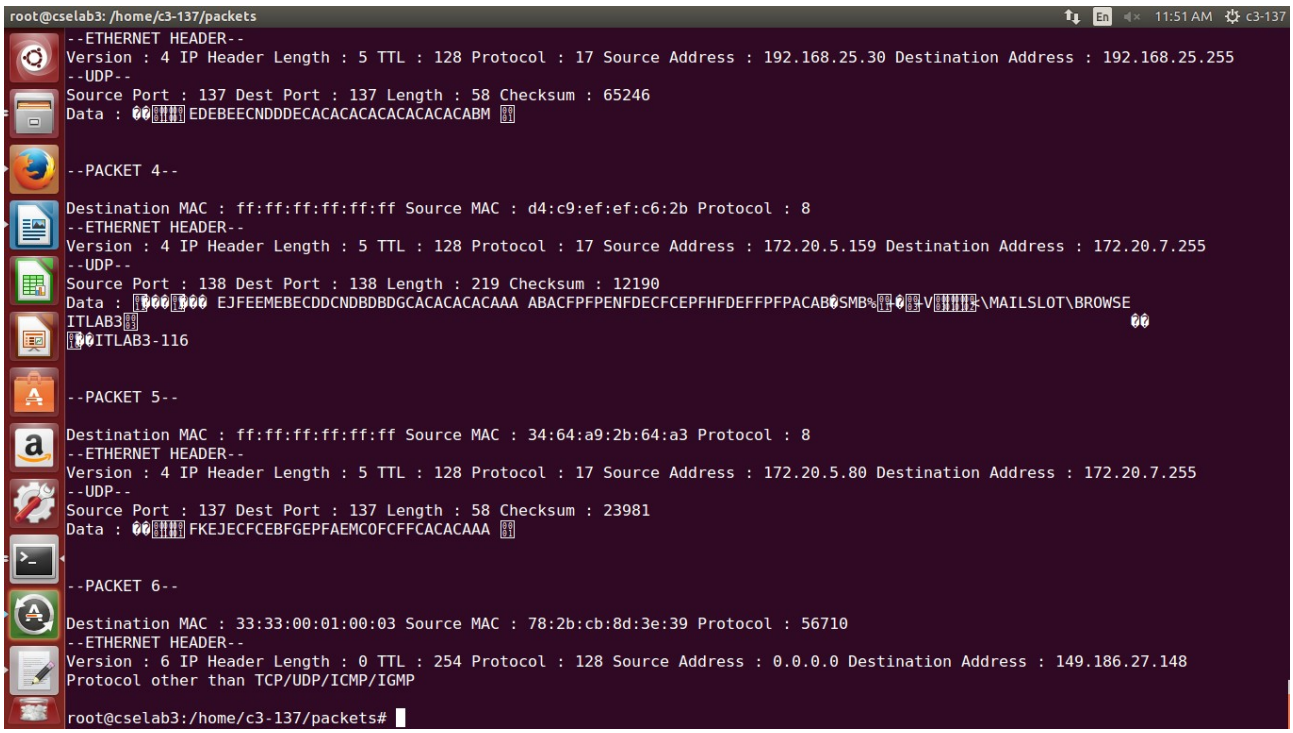
```
     iph = unpack('!BBHHHBBH4s4s' , ip_header)

     version_ihl = iph[0]
     version = version_ihl >> 4
     ihl = version_ihl & 0xF

     iph_length = ihl * 4

     ttl = iph[5]
     protocol = iph[6]
     s_addr = socket.inet_ntoa(iph[8]);
     d_addr = socket.inet_ntoa(iph[9]);

     print 'Version : ' + str(version) + ' IP Header Length : ' +
str(ihl) + ' TTL : ' + str(ttl) + ' Protocol : ' + str(protocol) +
' Source Address : ' + str(s_addr) + ' Destination Address : ' +
str(d_addr)

     #TCP protocol
     if protocol == 6 :
          print '--TCP--'
          t = iph_length + eth_length
          tcp_header = packet[t:t+20]

          #now unpack them :)
          tcph = unpack('!HHLLBBHHH' , tcp_header)

          source_port = tcph[0]
          dest_port = tcph[1]
          sequence = tcph[2]
          acknowledgement = tcph[3]
          doff_reserved = tcph[4]
          tcph_length = doff_reserved >> 4

          print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Sequence Number : ' + str(sequence) + '
Acknowledgement : ' + str(acknowledgement) + ' TCP header length :
' + str(tcph_length)

          h_size = eth_length + iph_length + tcph_length * 4
          data_size = len(packet) - h_size

          #get data from the packet
          data = packet[h_size:]

          print 'Data : ' + data

     #ICMP Packets
     elif protocol == 1 :
          print '--ICMP--'
          u = iph_length + eth_length
          icmph_length = 4
          icmp_header = packet[u:u+4]
```

```
          #now unpack them :)
          icmph = unpack('!BBH' , icmp_header)

          icmp_type = icmph[0]
          code = icmph[1]
          checksum = icmph[2]

          print 'Type : ' + str(icmp_type) + ' Code : ' + str(code)
+ ' Checksum : ' + str(checksum)

          h_size = eth_length + iph_length + icmph_length
          data_size = len(packet) - h_size

          #get data from the packet
          data = packet[h_size:]

          print 'Data : ' + data

     #UDP packets
     elif protocol == 17 :
          print '--UDP--'
          u = iph_length + eth_length
          udph_length = 8
          udp_header = packet[u:u+8]

          #now unpack them :)
          udph = unpack('!HHHH' , udp_header)

          source_port = udph[0]
          dest_port = udph[1]
          length = udph[2]
          checksum = udph[3]

          print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Length : ' + str(length) + ' Checksum : ' +
str(checksum)

          h_size = eth_length + iph_length + udph_length
          data_size = len(packet) - h_size

          #get data from the packet
          data = packet[h_size:]

          print 'Data : ' + data

     #IGMP packets
     elif protocol == 88 :
          print '--IGMP--'
          u = iph_length + eth_length
          igmph_length = 8
          igmp_header = packet[u:u+8]
```

```
        #now unpack them :)
        igmph = unpack('!HHHH' , igmp_header)

        source_port = igmph[0]
        dest_port = igmph[1]
        length = igmph[2]
        checksum = igmph[3]

        print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Length : ' + str(length) + ' Checksum : ' +
str(checksum)

        h_size = eth_length + iph_length + igmph_length
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data

    #some other IP packet
    else :
        print 'Protocol other than TCP/UDP/ICMP/IGMP'

    print
```

Output:





Task 3: Write a program to extract and print the TCP segment information including the header.

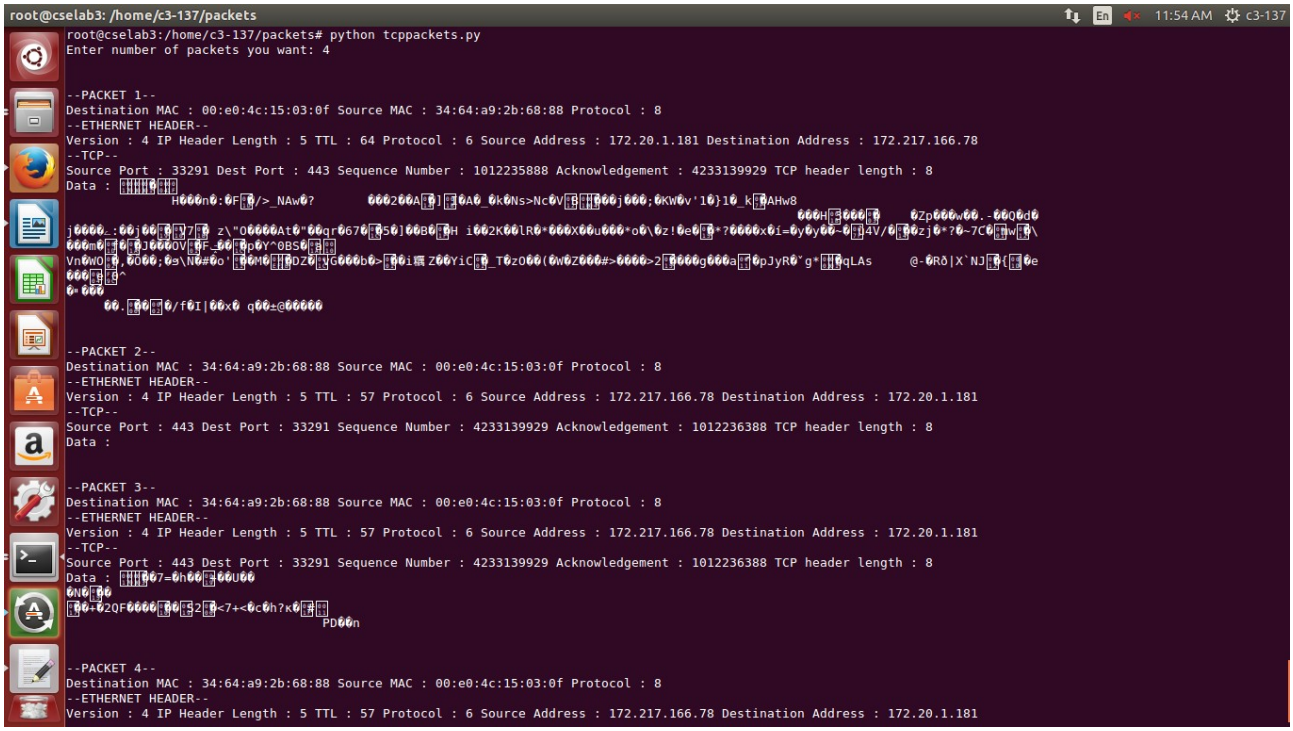
Program:

```
import socket, sys
from struct import *

#Convert a string of 6 characters of ethernet address into a dash
separated hex string
def eth_addr (a) :
     b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) ,
ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))
     return b

#create a AF_PACKET type raw socket (thats basically packet level)
#define ETH_P_ALL   0x0003          /* Every packet (be
careful!!!) */
try:
     s = socket.socket( socket.AF_PACKET , socket.SOCK_RAW ,
socket.ntohs(0x0003))
except socket.error , msg:
     print 'Socket could not be created. Error Code : ' +
str(msg[0]) + ' Message ' + msg[1]
     sys.exit()
n = input('Enter number of packets you want: ')
i = 0
# receive a packet
while(i<n):
     packet = s.recvfrom(65565)
```

```
     #packet string from tuple
     packet = packet[0]

     #parse ethernet header
     eth_length = 14

     eth_header = packet[:eth_length]
     eth = unpack('!6s6sH' , eth_header)
     eth_protocol = socket.ntohs(eth[2])

     #Parse IP header
     #take first 20 characters for the ip header
     ip_header = packet[eth_length:20+eth_length]

     #now unpack them
     iph = unpack('!BBHHHBBH4s4s' , ip_header)

     version_ihl = iph[0]
     version = version_ihl >> 4
     ihl = version_ihl & 0xF

     iph_length = ihl * 4

     ttl = iph[5]
     protocol = iph[6]
     s_addr = socket.inet_ntoa(iph[8]);
     d_addr = socket.inet_ntoa(iph[9]);


     #TCP protocol
     if protocol == 6 :
          i+=1
          print
          print
          print '--PACKET '+str(i)+'--'
          print 'Destination MAC : ' + eth_addr(packet[0:6]) + '
Source MAC : ' + eth_addr(packet[6:12]) + ' Protocol : ' +
str(eth_protocol)
          print '--ETHERNET HEADER--'
          print 'Version : ' + str(version) + ' IP Header Length :
' + str(ihl) + ' TTL : ' + str(ttl) + ' Protocol : ' +
str(protocol) + ' Source Address : ' + str(s_addr) + ' Destination
Address : ' + str(d_addr)
          t = iph_length + eth_length
          tcp_header = packet[t:t+20]

          #now unpack them :)
          tcph = unpack('!HHLLBBHHH' , tcp_header)

          source_port = tcph[0]
          dest_port = tcph[1]
          sequence = tcph[2]
          acknowledgement = tcph[3]
```

```
        doff_reserved = tcph[4]
        tcph_length = doff_reserved >> 4

        print '--TCP--'
        print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Sequence Number : ' + str(sequence) + '
Acknowledgement : ' + str(acknowledgement) + ' TCP header length :
' + str(tcph_length)

        h_size = eth_length + iph_length + tcph_length * 4
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data
```

Output:



Task 4: Write a program to extract and print the UDP segment information.

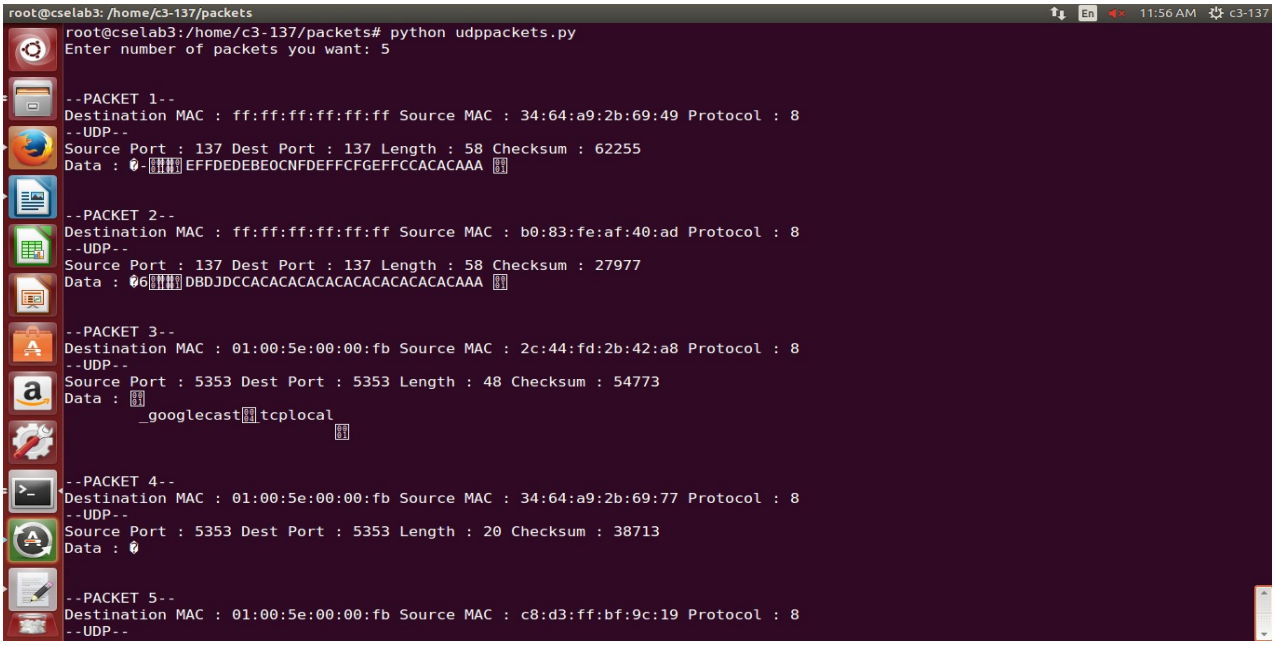
Program:

```
import socket, sys
from struct import *

#Convert a string of 6 characters of ethernet address into a dash
separated hex string
```

```
def eth_addr (a) :
     b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) ,
ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))
     return b

#create a AF_PACKET type raw socket
try:
     s = socket.socket( socket.AF_PACKET , socket.SOCK_RAW ,
socket.ntohs(0x0003))
except socket.error , msg:
     print 'Socket could not be created. Error Code : ' +
str(msg[0]) + ' Message ' + msg[1]
     sys.exit()
n = input('Enter number of packets you want: ')
i = 0
# receive a packet
while(i<n):
     packet = s.recvfrom(65565)

     #packet string from tuple
     packet = packet[0]

     #parse ethernet header
     eth_length = 14

     eth_header = packet[:eth_length]
     eth = unpack('!6s6sH' , eth_header)
     eth_protocol = socket.ntohs(eth[2])

     #Parse IP header
     #take first 20 characters for the ip header
     ip_header = packet[eth_length:20+eth_length]

     #now unpack them
     iph = unpack('!BBHHHBBH4s4s' , ip_header)

     version_ihl = iph[0]
     version = version_ihl >> 4
     ihl = version_ihl & 0xF

     iph_length = ihl * 4

     ttl = iph[5]
     protocol = iph[6]
     s_addr = socket.inet_ntoa(iph[8]);
     d_addr = socket.inet_ntoa(iph[9]);


     if protocol == 17 :
          i+=1
          print
          print
          print '--PACKET '+str(i)+'--'
```

```
        print 'Destination MAC : ' + eth_addr(packet[0:6]) + '
Source MAC : ' + eth_addr(packet[6:12]) + ' Protocol : ' +
str(eth_protocol)
        print '--UDP--'
        u = iph_length + eth_length
        udph_length = 8
        udp_header = packet[u:u+8]

        #now unpack them :)
        udph = unpack('!HHHH' , udp_header)

        source_port = udph[0]
        dest_port = udph[1]
        length = udph[2]
        checksum = udph[3]

        print 'Source Port : ' + str(source_port) + ' Dest Port :
' + str(dest_port) + ' Length : ' + str(length) + ' Checksum : ' +
str(checksum)

        h_size = eth_length + iph_length + udph_length
        data_size = len(packet) - h_size

        #get data from the packet
        data = packet[h_size:]

        print 'Data : ' + data
```

Output:



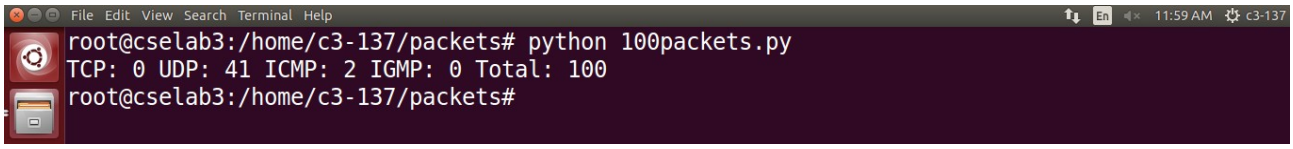
Task 5: Write a program that captures 100 packets and display the count of ICMP, IGMP, TCP, UDP, and IP packets.

Program:

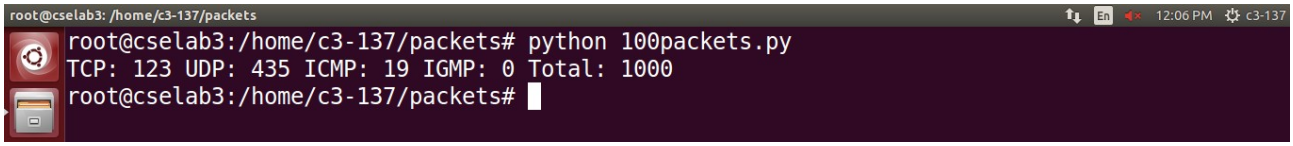
```
import socket, sys
from struct import *

#Convert a string of 6 characters of ethernet address into a dash
separated hex string
def eth_addr (a) :
     b = "%.2x:%.2x:%.2x:%.2x:%.2x:%.2x" % (ord(a[0]) , ord(a[1]) ,
ord(a[2]), ord(a[3]), ord(a[4]) , ord(a[5]))
     return b

#create a AF_PACKET type raw socket

try:
     s = socket.socket( socket.AF_PACKET , socket.SOCK_RAW ,
socket.ntohs(0x0003))
except socket.error , msg:
     print 'Socket could not be created. Error Code : ' +
str(msg[0]) + ' Message ' + msg[1]
     sys.exit()
# receive a packet
icmp = 0
tcp  = 0
udp  = 0
igmp = 0
for i in range(0, 1000):
     packet = s.recvfrom(65565)

     #packet string from tuple
     packet = packet[0]
     eth_length = 14

     #Parse IP header
     #take first 20 characters for the ip header
     ip_header = packet[eth_length:20+eth_length]

     #now unpack them
     iph = unpack('!BBHHHBBH4s4s' , ip_header)
     ttl = iph[5]
     protocol = iph[6]
     #TCP protocol
     if protocol == 6 :
          tcp+=1

     #ICMP Packets
     elif protocol == 1 :
          icmp+=1

     #UDP packets
```

```
     elif protocol == 17 :
          udp+=1

     #IGMP Packets
     elif protocol == 88 :
          igmp+=1
     sys.stdout.write("\rTCP: %d UDP: %d ICMP: %d IGMP: %d Total:
%d" %(tcp, udp, icmp, igmp, i+1))
     sys.stdout.flush()
print
```

Output:



For 1000 packets,



For 10,000 packets,