7.1 Select one UDP packet from your trace. From this packet, determine

1) how many fields there are in the UDP header.
2) the name of each fields in the UDP header.
3) the length (in bytes) of each fields in the UDP header.
4) What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 3) above)
5) What is the largest possible source port number? (Hint: same as the hint in 4) above.)
6) What is the protocol number for UDP?(Give your answer in both hexadecimal and decimal notation.)

7.2 Finish the question 4，6，7，9，10，12 of Wireshark_TCP_v7.0.pdf

7.3 RDT Implementation

In this assignment, you need our given UDP socket class to implement RDT protocol. Then implement a Echo Server and Client.

I. Requirement
   i. Your protocol needs to ensure the reliability of data transfer. Packet loss and payload corruption might happen.
      i. To deal with packet loss, using ack and retransmission according to GBN the textbook.
      ii. To deal with payload corruption, you need to design a checksum of your payload.
   ii. Your RDT protocol should be like TCP, which means it’s a stream-oriented protocol, not packet-oriented.

i. To establish a connection, you might need to do things like things in TCP:
   1. SYN
   2. SYN, ACK
   3. ACK

ii. To close a connection, you might need to do things like things in TCP:
   1. FIN
   2. ACK
   3. FIN
   4. ACK

iii. Payload

Your payload might be like this:

SYN	FIN	ACK	SEQ	SEQ ACK	LEN	CHEKCSUM	Payload
1 bit	1 bit	1 bit	4 byte	4 byte	4 byte	2 byte	LEN

i. Checksum Calculation Example(just for reference)

```
def calc_checksum(payload):
    sum = 0
    for byte in payload:
        sum += byte
    sum = -(sum % 256)
    return (sum & 0xFF)
```

II. API reference:

rdt code example:

```
from udp import UDPsocket # import provided class
class socket(UDPsocket):
    def __init__():
        super(socket, self).__init__()

    def connect():
        # send syn; receive syn, ack; send ack
        # your code here
        pass

    def accept():
        # receive syn; send syn, ack; receive ack
        # your code here
        pass

    def close():
        # send fin; receive ack; receive fin; send ack
        # your code here
        pass

    def recv():
        # your code here
        pass

    def send():
        # your code here
        pass
```

server code example:

```
from rdt import socket

server = socket()
server.bind((SERVER_ADDR, SERVER_PORT))
while True:
    conn, client = server.accept()
    while True:
        data = conn.recv(2048)
        if not data: break
            conn.send(data)
        conn.close()
```

client code example:

```
from rdt import socket

client = socket()
client.connect((SERVER_ADDR, SERVER_PORT))
client.send(MESSAGE)
data = client.recv(BUFFER_SIZE)
assert data == MESSAGE
client.close()
```