1. What does INADDR_ANY mean?

   INADDR_ANY is a constant in networking programming that represents the “any” address or “wildcard” address. It used to bind a socket to all available network interfaces of a machine, so that it can receive incoming network connections from any IP address.

   INADDR_ANY also means that a server application can listen for incoming connections on all available IP addresses of the local machine. Otherwise, the INADDR_ANY constant is usually defined as 0.0.0.0 in IPv4 network programming.
2. What’s the difference between bind() and listen()?

   bind() is used to specify the local address for the socket, while listen() is used to indicate that the socket is ready to receive incoming connections and to specify the maximum number of connections that can be queued up. bind() is usually called first to specify the local address and port number, followed by listen() to put the socket in a passive listen mode.
3. Usually, we set up the server’s port and exclude the client’s. Who determines the client’s port and what’s the benefit?

   The client’s port is determined by the operating system. The benefits of letting the operating system choose an ephemeral port number for the client is that it allows multiple client applications on the same machine to establish connections to the same server without having to coordinate with each other. That is, each client can use a different source port number for its connections, which allows server to distinguish between the different clients and send response packets back to the correct client.
4. What is little endian and big endian? Why do most network byte order use big endian?

   In a little endian system, the least significant byte of a multi-byte data type is stored at the lowest memory address, while the most significant byte is stored at the highest memory address. This means that the data type is read from right to left, with the least significant byte being read first.

   In a big endian system, the most significant byte of a multi-byte data type is stored at the lowest memory address, while the least significant byte is stored at the highest memory address. This means that the data type is read from left to right, with the most significant byte being read first.

   Most network bytes orders use big endian as the standard byte order for transmitting data over a network. This is because different computers may use different byte orders, and the big endian format provides a consistent, unambiguous way to represent data that can be easily understood by all systems.

5. Why do we need a pseudo header ?

   The purpose of a pseudo header is to include additional information in the checksum calculation that is not explicitly included in the header fields. Specifically, the pseudo header includes the source and destination IP address, the protocol number, and the length of the transport layer header and data. This allows the receiving system to verify that the data was current or not modified during transmission and that it was received from the expected source.

   By including the additional information in the pseudo header, the checksum can detect errors that might occur due to incorrect IP addresses, incorrect protocol number, or incorrect length of the transport layer data. Overall, the pseudo header can ensure that the reliability and security of data transmission in network protocols which rely on checksum and error detection.

6. For the code below, what's difference between client_fd and socket_fd ?

   A socket is an endpoint for sending or receiving data across a network. The socket is identified by a unique file descriptor (fd), which is an integer used to reference the socket in subsequent function calls.

   client_fd is the file descriptor associated with the newly created socket that is used to communicated with a specific client, while "socket_fd" is either the file descriptor associated with the server socket used for listening or the file descriptor associated with any socket used for sending or receiving data.

7. When using the send() function and recv() function, why do we not need the address?

   This is because a socket was construct, it was typically bound to a specific local address and port. When data was sent or received on a socket using send() and recv(), the operating system will automatically choose the same port and local address which the socket was bound to, so there is no need to specify it again.

   Similarly, when client socket connects to server, the client socket will be automatically assigned a random source port and the server socket will know the client's port and address based on the information in the incoming packets, so there is no need to specify the port and address information for the client.

8. Write about what you have learned from Lab 2.

   In Lab 2, I have learned network socket programing. I know the interaction between server and client and the setting of the socket of server and client. From building a TCP socket to sending packets between server and client, I can use coding skill to implement the things which have been taught in class. Otherwise, I also learn how to construct TCP header and pseudo header by using l4header information and l3header information. Further, using TCP header and pseudo header calculates checksum. To calculate checksum, we need to combine each 16bit in TCP header and pseudo header respectively, add them together, and then add the sum of TCP header and pseudo header. Wrapping around the sum, we can get the correct checksum.