1. **What is the maximum number of ports/sockets which can be open simultaneously on a server or a client (2 pts)**

single client cannot open more than 65535 simultaneous connections to a server. But a server can (theoretically) server 65535 simultaneous connections per client.

1. **Which sockets are considered reserved and should not be used by regular users/programmers (2 pts)**

should note that port numbers between 0 and 1,023 are reserved for privileged users (that is, super user or root).

1. **What is the main reason for closing a socket after finishing using it? (2 pts)**
   1. the close a critical part of protocol because without it the server don't know when the client is finished sending characters as example
   2. One way or another, if you don't close a socket, your program will leak a file descriptor. Programs can usually only open a limited number of file descriptors, so if this happens a lot, it may turn into a problem.
   3. If the socket is bound to an address, no other socket will be able to bind to the same address until the socket is closed. If it's bound to a well-known port (like 25, or 80, for example), this will prevent anything else from binding to that port... which would probably be the most serious problem you might face. If it's bound to an ephemeral port, this is not as important, but, still, there are a limited number of ports and if this happens a lot then you may run out. Note that UDP sockets can be used freely without ever being bound, so this problem may not apply to UDP sockets.
   4. If the socket is a listening socket (listen() has been called on it), the system will continue to accept connections on this socket if you don't close it (but only up to the backlog limited specified in listen()).
2. **What is the main difference between using TCP or UDP for socket programming? (2 pts)**
   1. In UDP, as you have read above, every time you send a datagram, you have to send the local descriptor and the socket address of the receiving socket along with it.
   2. Since TCP is a connection-oriented protocol, on the other hand, a connection must be established before communications between the pair of sockets start. So there is a connection setup time in TCP
   3. In UDP, there is a size limit of 64 kilobytes on datagrams you can send to a specified location,
   4. In TCP there is no limit. Once a connection is established, the pair of sockets behaves like streams: All available data are read immediately in the same order in which they are received.
   5. UDP is an unreliable protocol -- there is no guarantee that the datagrams you have sent will be received in the same order by the receiving socket.
   6. On the other hand, TCP is a reliable protocol; it is guaranteed that the packets you send will be received in the order in which they were sent.
   7. TCP is useful for implementing network services -- such as remote login (rlogin, telnet) and file transfer (FTP) -- which require data of indefinite length to be transferred.
   8. UDP is less complex and incurs fewer overheads. It is often used in implementing client/server applications in distributed systems built over local area networks.
3. **How does the server know the port number of the client (given that the client did not declare a port number???) (2 pts)**
   1. It's part of the TCP (or UDP, etc.) header, in the packet. So the server finds out because the client tells it. This is similar to how it finds out the client's IP address (which is part of the IP header).
   2. E.g., every TCP packet includes an IP header (with source IP, destination IP, and protocol [TCP], at least). Then there is a TCP header (with source and destination port, plus more).