4.

Correct answer-C

My answer-H

I assumed with SOCK_DGRAM that connect being called meant that the socket had chosen a destination, so that send to function wouldn't be used, meaning I marked C-connect(). But after accept is called, this is what sets the destination; and with this SOCK_DGRAM type, socket being called means send to function isn't necessary. So the answer is H-accept() and socket().

5.

Correct answer-G

My answer-A

SOCK_STREAM can call accept(). Any socket can call listen(), but I incorrectly believed only SOCK_DGRAM could, so I answered A-accept() when the correct answer of the question would be G-accept() and listen().

6.

Correct answer-B

My answer-D

I assumed that the listen() call had to be used with servers, since they are listening for connections, so I answered D-listen(), while I assumed that bind() was optional for both. But apparently, bind() is required for servers, but listen() is optional for servers, so the correct answer is B-bind().

9.

Correct answer-D

My answer-A

I assumed when accept() was called, the socket could call accept() again but that it couldn't be used to send and receive any more, but that after calling listen() it could receive data, and since I correctly thought calling socket() would still let it transmit data and receive it, so I answered A-accept(), but calling accept() actually lets it receive data from the connection it accepted, while listen() being called means it is only listening instead of transmitting data but it can still call accept(), so the correct answer is D-listen().

16.

Correct answer-J

My answer-A

I assumed that because there was no data in buf4, the call recv(sock, buf4, 50, 0) would return 0, so I answered A-0. But because there is no data, it actually blocks, so the answer is J-Blocks.

17.

Correct answer-I

My answer-H

I assumed that because the call was recv(sock, buf1, 160, 0) and there were 100 bytes sent, that 100 would be returned immediately, so I answered H-100, forgetting that with SOCK_STREAM, the bytes are streamed, so receive calls receive all bytes they can, and all bytes are basically in the same buffer, so because there's 300 bytes in that larger buffer, the receive call receives 160 bytes, so I-160 is the correct answer.

19.

Correct answer-E

My answer-F

I assumed that because the call was recv(sock, buf3, 70, 0) and there were 100 bytes sent, that 70 would be returned immediately, so I answered F-70, forgetting that with SOCK_STREAM, the bytes are streamed, so receive calls receive all bytes they can, and all bytes are basically in the same buffer, so because there's 300 bytes in that larger buffer at the start and 240 have already been read, there are 60 bytes left, so the receive call receives 60 bytes, so E-60 is the correct answer.

31.

Correct answer-C

My answer-A

I assumed that we should call recv() repeatedly to get all bytes, so I answered A-If nread>0, call recv() again repeatedly until nread==0. But the client actually has to know the number of bytes in the response, so the answer is C-If buf doesn't have "\r\n\r\n" call recv() again repeatedly until it does.

36.

Correct answer-A

My answer-B

I assumed that because there were many threads, and there was no return statement in the main function, it meant that the program would wait for its threads, so I answered B-false to the question about the process exiting before threads have finished. But there was no join call, and main will automatically return if no return statement is there, and so that means that the threads might not be done by then and main function won't wait for them, so the answer A-true instead.