1. Question:
TCP/IP checksum algorithm seems pretty clear. But, their range is somewhat vague. Is data range is restricted to payload only or they include op, shift, and total length?
Answer:
In TCP/IP RFC (RFC 793), the checksum field is defined as following.
"The checksum field is the 16 bit one's complement of the one's complement sum of all 16-bit words in the header and text"
Our checksum field is exactly same. It should be calculated for all header fields (operation, shift, length) and data.

2. Question:
Does 10M limit include header(op+key+checksum+length = 16 Byte, thus maximum size of string is 10M - 16B) or not?
Answer:
10MB includes the header. (Tip: the value of length field includes the size of header.)

3. Question:
When should the client send message to server? Like, when they meet new line from the input string, the input string has accumulated up to 9984 bytes, ...
Answer:
The way you split the string received from stdin is up to you. Just make sure the size of a message does not exceed 10MB.

4. Question:
In client, is it ok to assume input is "printable string + '\n'" * many times?
specifically, does input file consist of (([A-Za-z0-9!"#$%&'()*+,.\\/:;<=>?@ ^-`{|}~
\t-])*\n)+ in regular expression?
Answer:
Consider input file is a regular text file. And, the file has an EOF(end-of-file) at the end.
(https://en.wikipedia.org/wiki/End-of-file)
When you redirect a text file to stdin (i.e., ./client < sample.txt), EOF is also streamed into stdin.
When you write a string to stdin, you can signal EOF with ctrl+D in linux system.
HINT: Whatever function you use for reading a text from stdin, checking whether input string has EOF would be helpful.

5. Question:
Can we assume that server's IP is IPv4? (since we are testing the codes up on a specific server?)
Or should we also consider clients connecting to IPv6?
Answer:
You can assume ipv4

6. Question:
Why do we use big endian for each fields? Shouldn't the entire message be big endian?
Answer:
Before answering the question, I want you to be clear on the concept of network protocol header and endian. On one hand, network protocol header, such as our {op, shift, checksum, length}, specifies how we should interpret each part (field) of a message. For example, our header specifies that the very first byte represents the kind of operation to be performed. On the other hand, endian specifies which byte is the least significant byte (LSB) and which byte is the most significant byte (MSB) in a multi-byte variable. For example, big-endian specifies that the first byte is MSB.
In summary, the network protocol is for a message and endian is for a multi-byte variable.

7. Question:
Why we should not use ntohs() or ntohs() for checksum?
Answer:
Checksum is for error checking, and its mathematical value is not important. (In addition, checksum calculation does not care endian due to wrap-around logic)

8. Question:
The buffer of server should be at least 10MB so I declared it like "char buf[10*1024*1024];".
But I think it can exceed memory capacity when many processes are created at the same time.
Do we need to manage memory using dynamic allocation?
Answer:
You should dynamically allocate the buffer.
As you may have learned in previous classes (maybe OS?), if you declare a static array (as you mentioned) then it is allocated in stack.
Typically, Linux kernel has default stack size limit for each process, and it was 8MB for me (Ubuntu 16.04, Linux 3.16).
So 10 MB buffer should be allocated in heap (by malloc()).
You can check your own stack size limit by "ulimit -s" command which returns the stack size limit in KiB)

9. Question:
My client works fine with short messages, however the test server cuts connections when message size is large (several megabytes).
Answer:
For a TCP packet, the packet size (more specifically, IP fragment size) cannot exceed 65535 bytes.
(And the typical maximum packet size is 1514).
Most TCP checksum calculation codes on the Internet assume this. They usually calculate the checksum in three separate steps, 1. summation, 2. wrap-around, 3. inversion. In the summation step, they use 32-bit variable (unsigned, unsigned int, or uint32-t). This is okay since 32 bits (precisely, 31 bits) are sufficient for summing any 32768 (= 65535 / 2) unsigned 16-bit variables.
However, in our case, the message size can be 10M which is much larger than 65535. This demands you to update your checksum implementation.

10. Question:
I was struggled with the checksum field, since I considered it as a big-endian.
Now I know that the test server will reply my messages only if the checksum is not packed by htons, but still I can't reason it fully why it works. Why the checksum field doesn't care about endian-ness, even it is 2- bytes long?
Answer:
You may want to read RFC 1071, Computing the Internet Checksum.
See (B) Byte Order Independence at page 3. Link: https://tools.ietf.org/html/rfc1071

11. Question:
Should the client validate checksum and the length part of the response from the server?
Answer:
Yes, the client should.

12. Question:
Do we have a specific format for stdout outputs for server and client?
Can we have informative messages like "connecting to : 127.0.0.0" or "terminating - client disconnected," for example?
Answer:
For stdout,
1. your client should never print any message except the result of encryption/decryption.
2. your server should not print anything.
You can use stderr for your own debugging messages.

13. Question:
I wrote some codes in files 'protocol.c/h', which is not in the submission format. But will it affect the evaluation process even if I handle it with proper Makefile ?
Answer:
It is okay if
1. Your Makefile works fine,
2. and additional files (for your case, protocol.c/h) are written by you or have proper references.

Splitting your source code into multiple files is a good habit :)

14. Question:
How can I test my server with multiple client?
I can do it one or two client by open more shell, but I am not sure.
Is there any efficient way to test my server with multiple client?
Answer:
Maybe trying simple shell script will work.
[How to run multiple process in bash] http://stackoverflow.com/questions/5238103/how-to-start-multipleprocesses-in-bash

15. Question:
Should we move to the next key character even when we meet non-alphabetic characters?
Answer:
No. For example, encrypting "aa!a" with key of "abcd" results "ab!c" (not "ab!d".)