IRC RFC 1459 Questions:

1.What uniquely distinguishes clients on a server?

A client is anything connecting to a server that is not another server. **Each client on a server is distinguished by their unique nickname (having a maximum of 9 characters)**. (1.2 Clients)

2.What information must all servers have relating to users?

Along with their nickname, the servers have the following information pertaining to their users: the real name of the host the client is running on, the username of the client on the host and which server the client is connected to. (1.2 Clients)

3.How many bits are used in the protocol to define a character?

The protocol is based on a set of codes composed of **8** bits (octet). Messages are composed of any number of octets. (2.2 Character codes)

4.What is the binary representation of the character that is used to separate commands and parameters?

IRC messages are made of three parts, prefix (optional), the command the the command parameters. They are all separated by one or more ASCII space character(s) (0x20), hex to binary is **0010 0000** . (2.3 Messages)

5.What is/are the character(s) that mark(s) the termination of an IRC message?

Messages are lines of characters that always ends with a **CR-LF (carriage return - line feed) pair. CR and LF are control characters and are respectively coded, ASCII code 13 0x0D (\r) and ASCII code 10 0x0A (\n)**, and are used to mark the end of line. (2.3.1)

6.What's the maximum length of a message?

The maximum length of a message is 512 characters (including 2 reserved for CR-LF) for the command and its parameters. (2.3 Messages)

7.What is the only valid prefix that a client can put in a message?

Normally, clients should not use prefixes. However if they do, the only valid one to put is the registered nickname associated with the client. (2.3 Messages)

8.What does a server do when it receives a numeric reply from a client?

Numeric replies are not allowed to come from clients, thus, messages originating from clients to servers are silently dropped. (2.4 Numeric replies)

9.What is the content of the reply that a server generates when it receives a NICK message from a client that causes a nickname collision?
A NICK message is used to give a user a nickname or change its previous one. If one arrives in the server where an identical nickname for another client occurs, this results in a nickname collision. When this happens, all instances of the nickname are removed from the server database, then a KILL command is issued to remove the nickname from all the other server's databases. This essentially wipes the old nickname from the system. Numeric replies are :

	ERR_NONICKNAMEGIVEN	ERR_ERRONEUSNICKNAME
	ERR_NICKNAMEINUSE	ERR_NICKCOLLISION

(4.1.2 Nick message)

10.Name one security issue with the protocol? Refer to the relevant section in the RFC 1459 or its updates.
Privacy is a big security consideration, by having your data and identity compromised. Security considerations are considered in:
4.1 Connection Registration
All server connections should have a password, so that it will give security to the actual connection (prevent others from compromising).
4.1.1 Password message
The PASS command should be used to set a connection password and should be used before any attempt to register to send NICK/USER combinations. Servers have to do the same before any SERVER command.
4.1.3 User message
The hostname and server names are normally ignored by the IRC server when the USER command is used from a client that is directly connected to the server, this is mainly for security reasons. It is usually very easy for clients to lie about their username by replying only to the USER message, so using an "Identity Server" which helps set their real usernames.
5.5 Users
USERS command returns the list of users logged into the server, however due to privacy concerns, some servers have disabled the use of this command to protect their client's privacy.
7 Client and server authentication
Clients and servers can both have the same level of authentication. As such, IP address and hostname lookup (and reverse checks on them) can be performed on all connections made to the server. This can lead to privacy issues and people tracking where other clients are. However, it is possible to enforce password checks as a security measure.

References:
RFC 1459 - Internet Relay Chat Protocol, J. Oikarinen, D. Reed, May 1993, https://tools.ietf.org/html/rfc1459

Packet Analysis of IRC Connection Questions

1.Find the destination port that the client connected to on freenode.net

Destination Port TCP: 6667
IPv4: 38.229.70.22

2.Find the nickname that the session is using on the server.

NICK asf91

3.Find the period of time that the user is connected to the server

<u>When the user is connected to the server</u>
Response: :card.freenode.net 001 asf91 :Welcome to the freenode Internet Relay Chat Network asf91
[Time since first frame in this TCP stream: 6.582162531 seconds]

<u>When the user quits the server</u>
Response: :asf91!~tata4@144.172.195.176 QUIT :Quit: Leaving
[Time since first frame in this TCP stream: 541.879959101 seconds]

Total time: 535.29779657 seconds ~ 535 seconds

4.Find the last message that the user sent to the channel

Last private message:
PRIVMSG #tmp-445 :Find a special url using 'urlencoded-form' in the WS filter

5.Find the connection password

PASS plzUseEncrypt

Design description

Executions instructions are on README.md

Structure:

- common.py includes the common methods and constant across the app.
- irc_server.py is the server of the IRC.
- irc_client.py is the client of the IRC.

Design:

Server

- The server starts by binding host and port according to the provided configurations (command line arguments) in nonblocking mode. Then it uses select() to manage readable sockets. Upon a socket found in ready_to_read list, the server handles by calling handle_data() to process the received message or establish a new socket (for the new connection request if server_socket is found in ready_to_read).
- In handle_data(), the server processes the received message according to RFC protocol.
- broadcast() is used to send messages from the server to all of its clients via socket (exclude the sender and the server_socket).
- Information for non-blocking-sockets

Client

- The client receives arguments (server, port, nickname, username) from the command line and establishes a connection to the server. It then proceeds to register an user profile with the server using NICK and USER commands of RFC protocol. If the nickname already exists in the server, the client will receive a status of ERR_NICKNAMEINUSE and terminate.
- The client then joins a channel (default #global).
- After registering with the server successfully, the client can send messages to the server using the PRIVMSG command. The server will extract the message and broadcast server-wide to other connected clients at the moment.
- For receiving messages, the client creates a new thread to continuously listen for any data coming back from the server and put it up for display in the TUI.
- When a client wants to quit, type \quit in the chat will issue a QUIT command to the server. The server handles the command by closing the socket and deleting the user profile. Client also closes its socket before terminating.