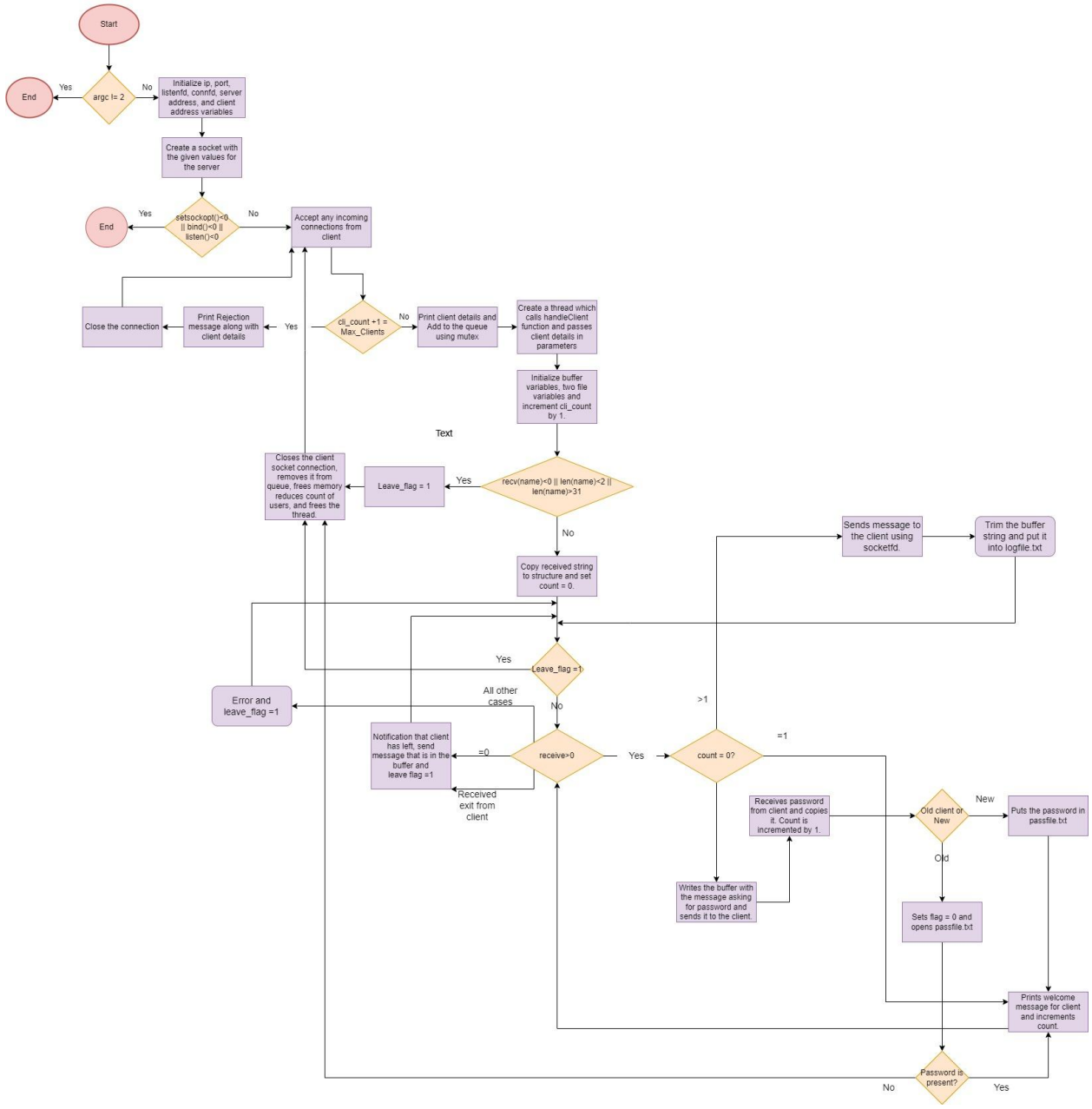
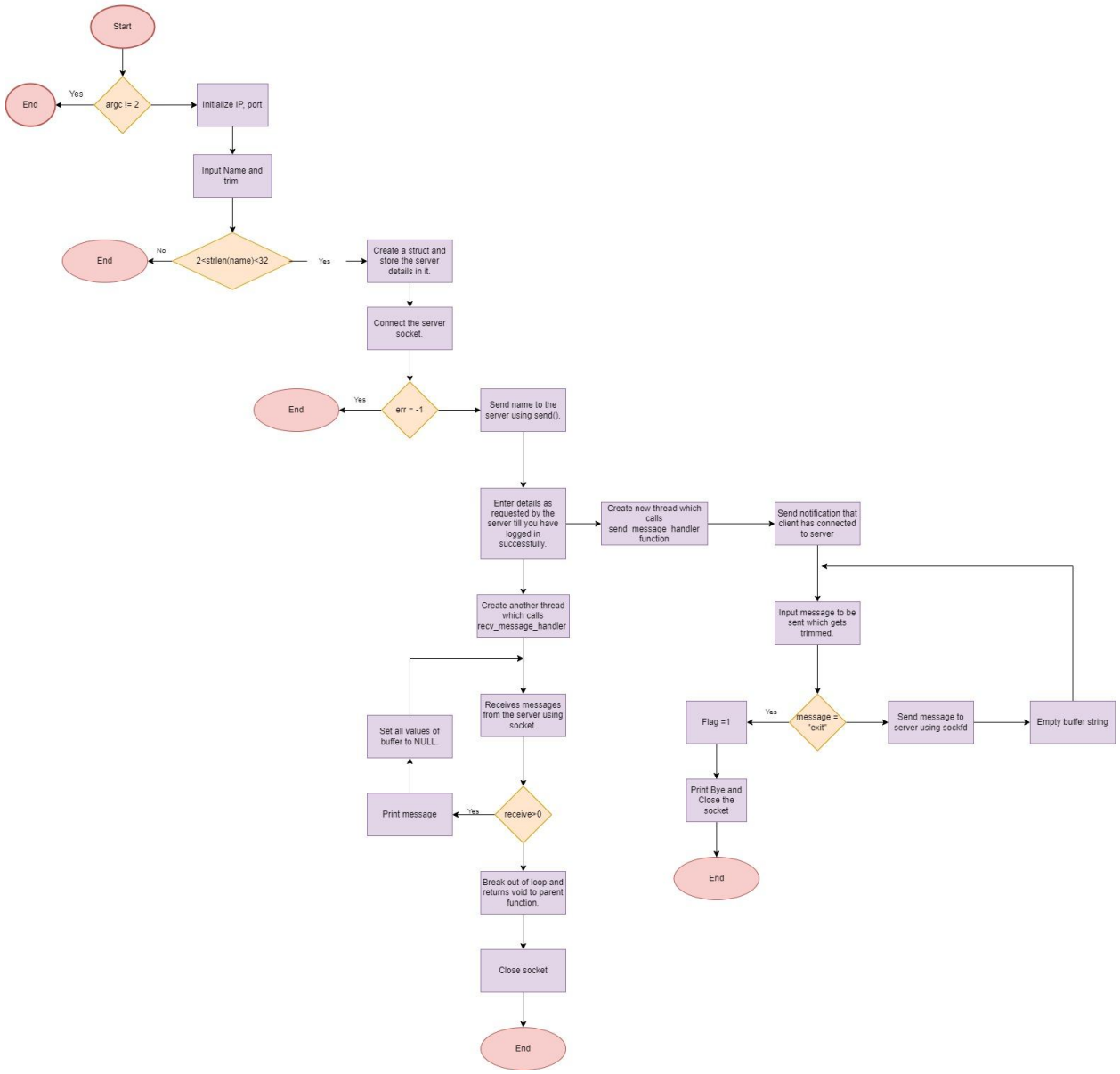
Server :


Start
End
Yes
argc != 2
No
Initialize ip, port, listenfd, connfd, server address, and client address variables
Create a socket with the given values for the server
End
Yes
setsockopt()<0 || bind()<0 || listen()<0
No
Accept any incoming connections from client
Close the connection
Print Rejection message along with client details
Yes
cli_count +1 = Max_Clients
No
Print client details and Add to the queue using mutex
Create a thread which calls handleClient function and passes client details in parameters
Initialize buffer variables, two file variables and increment cli_count by 1.
Text
Closes the client socket connection, removes it from queue, frees memory reduces count of users, and frees the thread.
Leave_flag = 1
Yes
recv(name)<0 || len(name)<2 || len(name)>31
No
Copy received string to structure and set count = 0.
Sends message to the client using socketfd.
Trim the buffer string and put it into logfile.txt
Yes
Leave_flag =1
Error and leave_flag =1
All other cases
No
>1
=1
Notification that client has left, send message that is in the buffer and leave flag =1
=0
receive>0
Yes
count = 0?
Received exit from client
Receives password from client and copies it. Count is incremented by 1.
Old client or New
New
Puts the password in passfile.txt
Old
Writes the buffer with the message asking for password and sends it to the client.
Sets flag = 0 and opens passfile.txt
Prints welcome message for client and increments count.
Password is present?
No
Yes

Client :


Start
End
Yes
argc != 2
Initialize IP, port
Input Name and trim
End
No
2<strlen(name)<32
Yes
Create a struct and store the server details in it.
Connect the server socket.
End
Yes
err = -1
Send name to the server using send().
Enter details as requested by the server till you have logged in successfully.
Create new thread which calls send_message_handler function
Send notification that client has connected to server
Create another thread which calls recv_message_handler
Input message to be sent which gets trimmed.
Receives messages from the server using socket.
Set all values of buffer to NULL.
Print message
Yes
receive>0
Flag =1
Yes
message = "exit"
Send message to server using sockfd
Empty buffer string
Print Bye and Close the socket
End
Break out of loop and returns void to parent function.
Close socket
End