The chat room application was implemented iteratively using the select() function. Multithreading was not used to handle multiple clients. I created a chat room struct that had a room name and port number as its attributes. I defined two arrays of size MAX_CLIENTS = 256: client_socket contains the file descriptors of all the client sockets, and client_room_name contains the room name of the chat room that each client is in. Similarly, I defined an array of size MAX_CHAT_ROOM = 25 to contain the file descriptors of all the chat room sockets. I also used the map data structure to create a pairing of a chat room file descriptor with its associated chat room struct.

The server uses two types of sockets: a main server socket and a master socket for each chat room that communicates with all clients in the chat room. Note that a client only communicates with the main server when it is not in a chat room. The client_socket array contains the file descriptor of the client regardless if it is communicating with the server or a chat room.

Inside the main function, I initialized the server socket by binding it to port 8080 and listening for new connections. I then declared and initialized all the data structures listed above as well as the fd_set to create a set of socket file descriptors that will be used for select.

Inside an infinite while loop, clear the socket set read_fd_set. Add the server, clients, and chat room file descriptors to read_fd_set. Find the maximum file descriptor in this set. Call the select() function to wait for activity on any of the file descriptors. If activity occurred on the server socket, then accept the incoming connection and add the new file descriptor to client_socket. If activity occurred on a chat room socket, then accept the incoming connection and add the new file descriptor to client_socket. Also update client_room_name with the name of the chat room receiving the connection. Otherwise, an operation occurred on a client socket. Check each client socket that is set and read in the message. If the read operation failed, close the client socket and reset the client file descriptor and room name. Otherwise, the read operation was successful. Parse the first word of the message. If the first word (capitalized) is “CREATE”, “DELETE”, “LIST”, or “JOIN”, then perform the necessary operations for the command and send a response back to the client. Otherwise, the client message was designated for a chat room. Check if the client is in a chat room, and then send the message to all other clients in the same chat room by verifying that the elements in client_room_name are equal.

Note that clients disconnecting is handled by checking if the read operation on that file descriptor failed.