Command line/Usage:

Compile using simple gcc.
e.g. gcc udp_ind.c -o <output>

To run you need to give input file as command line argument.

eg ./chat <input-file>

Some example input files are 'client1' 'client2'
Output is on terminal.

When running normal typing will send messages with 2 exceptions. If you type '%' in beginning of a line, it changes status message to whatever follows. If you type '!' , then it will resend the arrival message to all on your list. (A kind of manual override).

Input File Structure:

Input file has the following structure.
• The first line has an integer telling the number of hosts that follow.(let it be n)
• The next n lines have the IP addresses of the nodes
• The next line has the initial status message for this node
• The next line has an integer which is 0 or 1 depending on whether we must wait till all hosts are connected to start chatting.

Command Effect

Command	Effect
%<text>\n	The status message changed to <text>
!	The initial header send again to everyone
<text>\n	The <text> sent as chat message

Protocol:

The protocol is easily understandable in terms of what are the headers, when are they sent and what they mean.

Header	Format	Meaning	Event when it is send
r	r<text>	This header announces arrival of a node for chat. <text> is the status message of the sending node. It is sent in many ways to ensure reliability of being connected e.g. it is sent after every 5 messages of 'd' or 's' type.	(1) When the program starts, this message is sent to each node read in input file. (2) After every 5 messages of 'd' or 's' type, it is sent to every node currently in chat (As known to this node). (3) Whenever user inputs !, it is sent in the same way as in (2).
a	a<text>	This is the reply header to r. It creates a 2-way handshake. <text> is the status message of the sender.	It is sent as a reply to an 'r' message
s	s%<text>	This is status message change header. <text> is the status message.	It is sent when user changes status message.
x	x	This is a reply to 'd' type message.	It is sent when the node receives the 'd' type message as a form of acknowledgement
n	n	This is a ready message	It is sent only when there is a condition that all the nodes on list must connect before chatting starts. It signifies that the node sending it is connected to everyone in its list.
d	d<text>	<text> is the chat message	It is sent during normal chat. IT is sent to everyone on the list known to be not dead and ready.

Other features:

(1) To detect whether a node is down:
When we send a 'd' type message to everyone , we put a mark against everyone's IP in out list.
We remove that integer only when we receive a 'x' type message. If by the time we send another
'd' type message the node has not responded, we mark the node as dead.
(2) An accidentally downed node can reconnect to an existing chat
(3) Limitation: It does not work on localhost, it works on a network only.(As all programs run on
the same port.)
(4) Limitation: There is no reliability in message delivery as UDP protocol is used.