1. Write a Program to demonstrate the concept of using socket to communicate between server and client.

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void error(const char * msg) {
        perror(msg);
        exit(1);
}
int main(int argc, char * argv[]) {
        int sockfd, newsockfd, portno;
        socklen_t clilen;
        char buffer[256];
        struct sockaddr_in serv_addr, cli_addr;
        int n;
        if (argc < 2) {
                fprintf(stderr, "ERROR, no port provided\n");
                exit(1);
        }
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0)
                error("ERROR opening socket");
        bzero((char * ) & serv_addr, sizeof(serv_addr));
        portno = atoi(argv[1]);
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons(portno);
        if (bind(sockfd, (struct sockaddr * ) & serv_addr,sizeof(serv_addr)) < 0)
                error("ERROR on binding");
        listen(sockfd, 5);
        clilen = sizeof(cli_addr);
        newsockfd = accept(sockfd,(struct sockaddr * ) & cli_addr, &clilen);
        if (newsockfd < 0)
                error("ERROR on accept");
        bzero(buffer, 256);
        n = read(newsockfd, buffer, 255);
        if (n < 0) error("ERROR reading from socket");
                printf("Here is the message: %s\n", buffer);
        n = write(newsockfd, "I got your message", 18);
        if (n < 0) error("ERROR writing to socket");
                close(newsockfd);
        close(sockfd);
        return 0;
}
```

client.c

```
#include   <stdio.h>
#include  <stdlib.h>
#include <unistd.h>
#include <string.h>
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
void error(const char *msg){
	perror(msg);
	exit(0);
}
int main(int argc, char *argv[]){
	int sockfd, portno, n;
	struct sockaddr_in serv_addr;
	struct hostent *server;
	char buffer[256];
	if (argc < 3) {
		fprintf(stderr,"usage %s hostname port\n", argv[0]);
		exit(0);
	}
	portno = atoi(argv[2]);
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	server = gethostbyname(argv[1]);
	if (server == NULL) {
		fprintf(stderr,"ERROR, no such host\n");
		exit(0);
	}
	bzero((char *) &serv_addr, sizeof(serv_addr));
	serv_addr.sin_family = AF_INET;
	bcopy((char *)server->h_addr,(char *)&serv_addr.sin_addr.s_addr,server->h_length);
	serv_addr.sin_port = htons(portno);
	if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
		error("ERROR connecting");
	printf("Please enter the message: ");
	bzero(buffer,256);
	fgets(buffer,255,stdin);
	n = write(sockfd,buffer,strlen(buffer));/
	if (n < 0)
		error("ERROR writing to socket");
	bzero(buffer,256);
	n = read(sockfd,buffer,255);
	if (n < 0)
		error("ERROR reading from socket");
	printf("%s\n",buffer);
	close(sockfd);
	return 0;
}
```

Unknown Functions:

- **perror(string):** The "perror(msg)" function in C is used to print an error message to the standard error stream (stderr) along with a descriptive error message based on the value of the "errno" variable.
- **socket(AF_INET, SOCK_STREAM, 0):** The "socket(AF_INET, SOCK_STREAM, 0)" function in C is used to create a new communication endpoint (socket) for a network connection using the Internet Protocol version 4 (IPv4) address family and the Transmission Control Protocol (TCP) for reliable, connection-oriented data transmission.

- **bzero((char *) & serv_addr, sizeof(serv_addr)):** The "bzero((char *) & serv_addr, sizeof(serv_addr))" function in C is used to set the memory block starting from the address of "serv_addr" to zero, where "serv_addr" is a struct that represents a server's address.
- **htons(portno):** The "htons(portno)" function in C is used to convert a 16-bit integer value "portno" from host byte order (which is the byte order used by the CPU) to network byte order (which is the byte order used by the network).
- **bind(sockfd, (struct sockaddr *) & serv_addr,sizeof(serv_addr)):** The "bind(sockfd, (struct sockaddr *) & serv_addr, sizeof(serv_addr))" function in C is used to associate a socket file descriptor "sockfd" with a specific network address specified in the "serv_addr" structure. The "serv_addr" structure should contain the network address and port number that the socket should bind to.
- **listen(sockfd, 5):** The "listen(sockfd, 5)" function in C is used to make a bound socket, specified by the socket file descriptor "sockfd", a passive socket, which means that it can accept incoming connections. The second parameter "5" specifies the maximum length of the queue of pending connections that the socket can handle.
- **accept(sockfd,(struct sockaddr *) & cli_addr, &clilen):** The "accept(sockfd, (struct sockaddr *) & cli_addr, &clilen)" function in C is used to accept a new incoming connection on a socket, specified by the socket file descriptor "sockfd". The function blocks the execution of the program until a connection request arrives. Upon a successful connection request, the function returns a new socket file descriptor that can be used to communicate with the client.
- **read(newsockfd, buffer, 255):** The "read(newsockfd, buffer, 255)" function in C is used to read data from the socket specified by the file descriptor "newsockfd" into the buffer array, with a maximum length of 255 bytes. The function blocks the program's execution until data is received from the socket.
- **write(newsockfd, "I got your message", 18):** The "write(newsockfd, "I got your message", 18)" function in C is used to write data to the socket specified by the file descriptor "newsockfd". The function sends the string "I got your message" of length 18 bytes to the connected client through the socket.
- **gethostbyname(argv[1]):** The "gethostbyname(argv[1])" function in C is used to retrieve information about a host's address, specified as a string in the "argv[1]" parameter. The function returns a pointer to a "hostent" structure that contains information about the host, such as its name, IP address, and address type.
- **bcopy((char *)server->h_addr,(char *)&serv_addr.sin_addr.s_addr,server->h_length):** The "bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length)" function in C is used to copy the network address of a server, specified by the "server" pointer, to the "serv_addr" structure's "sin_addr.s_addr" field.

2. Demonstrate a program to continuously send messages between the client and the server.

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void error(const char *msg){
        perror(msg);
        exit(1);
}
int main(int argc, char *argv[]){
        int sockfd, newsockfd, portno;
        socklen_t clilen;
        char buffer[256];
        struct sockaddr_in serv_addr, cli_addr;
        int n;
        if (argc < 2) {
                fprintf(stderr,"ERROR, no port provided\n");
```

```
		exit(1);
	}
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	bzero((char *) &serv_addr, sizeof(serv_addr));
	portno = atoi(argv[1]);
	serv_addr.sin_family = AF_INET;
	serv_addr.sin_addr.s_addr = INADDR_ANY;
	serv_addr.sin_port = htons(portno);
	if (bind(sockfd, (struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
		error("ERROR on binding");
	listen(sockfd,5);
	clilen = sizeof(cli_addr);
	newsockfd = accept(sockfd,
	(struct sockaddr *) &cli_addr, &clilen);
	if (newsockfd < 0)
		error("ERROR on accept");
	char message[256];
	while(1){
		bzero(buffer,256);
		n = read(newsockfd,buffer,255);
		if(buffer[0]=='!') break;
		if (n < 0) error("ERROR reading from socket");
			printf("Message from client: %s\n",buffer);
		bzero(buffer,256);
		printf("Enter message:");
		fgets(buffer,255,stdin);
		if(buffer[0]=='!') break;
			n = write(newsockfd,buffer,18);
		if (n < 0) error("ERROR writing to socket");
	}
	close(newsockfd);
	close(sockfd);
	return 0;
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
void error(const char *msg){
	perror(msg);
	exit(0);
}
int main(int argc, char *argv[]){
	int sockfd, portno, n;
	struct sockaddr_in serv_addr;
	struct hostent *server;
	char buffer[256];
	if (argc < 3) {
		fprintf(stderr,"usage %s hostname port\n", argv[0]);
```

```
		exit(0);
	}
	portno = atoi(argv[2]);
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	server = gethostbyname(argv[1]);
	if (server == NULL) {
		fprintf(stderr,"ERROR, no such host\n");
		exit(0);
	}
	bzero((char *) &serv_addr, sizeof(serv_addr));
	serv_addr.sin_family = AF_INET;
	bcopy((char *)server->h_addr,(char *)&serv_addr.sin_addr.s_addr,
	server->h_length);
	serv_addr.sin_port = htons(portno);
	if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
		error("ERROR connecting");
	while(1){
		bzero(buffer,256);
		printf("Enter message:");
		fgets(buffer,255,stdin);
		n = write(sockfd,buffer,strlen(buffer));
		if(buffer[0]=='!') break;
		if (n < 0)
			error("ERROR writing to socket");
		bzero(buffer,256);
		n = read(sockfd,buffer,255);
		if(buffer[0]=='!')		break;
		if (n < 0)		error("ERROR reading from socket");
		printf("Message from server:%s\n",buffer);
	}
	close(sockfd);
	return 0;
}
```

3. Write a program to communicate with multiple clients at one time.

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <pthread.h>
#define MAX_CLIENTS 10
void error(const char * msg) {
	perror(msg);
	exit(1);
}
void * client_handler(void * arg) {
	int sockfd = * (int * ) arg;
	char buffer[256];
	int n;
	while (1) {
		bzero(buffer, 256);
```

```
                n = read(sockfd, buffer, 255);
                if (n < 0) error("ERROR reading from socket");
                if (buffer[0] == '!') break;
                        printf("Message from client: %s\n", buffer);
                bzero(buffer, 256);
                printf("Enter message:");
                fgets(buffer, 255, stdin);
                if (buffer[0] == '!') break;
                        n = write(sockfd, buffer, 18);
                 if (n < 0) error("ERROR writing to socket");
        }
        close(sockfd);
        pthread_exit(NULL);
}
int main(int argc, char * argv[]) {
        int sockfd, newsockfd, portno;
        socklen_t clilen;
        char buffer[256];
        struct sockaddr_in serv_addr, cli_addr;
        int n;
        if (argc < 2) {
                fprintf(stderr, "ERROR, no port provided\n");
                exit(1);
        }
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0)
                error("ERROR opening socket");
        bzero((char * ) & serv_addr, sizeof(serv_addr));
        portno = atoi(argv[1]);
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons(portno);
        if (bind(sockfd, (struct sockaddr * ) & serv_addr, sizeof(serv_addr)) < 0)
                error("ERROR on binding");
        listen(sockfd, MAX_CLIENTS);
        pthread_t threads[MAX_CLIENTS]
        int thread_count = 0;
        while (1) {
                clilen = sizeof(cli_addr);
                newsockfd = accept(sockfd, (struct sockaddr * ) & cli_addr, & clilen);
                if (newsockfd < 0)
                        error("ERROR on accept");
                printf("Client connected\n");
                pthread_create( & threads[thread_count], NULL, client_handler, & newsockfd);
                thread_count++;
                if (thread_count == MAX_CLIENTS) {
                        printf("Maximum number of clients reached\n");
                        break;
                }
        }
        for (int i = 0; i < thread_count; i++) {
                pthread_join(threads[i], NULL);
        }
        close(sockfd);
        return 0;
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
void error(const char * msg) {
	perror(msg);
	exit(0);
}
int main(int argc, char * argv[]) {
	int sockfd, portno, n;
	struct sockaddr_in serv_addr;
	struct hostent * server;
	char buffer[256];
	if (argc < 3) {
		fprintf(stderr, "usage %s hostname port\n", argv[0]);
		exit(0);
	}
	portno = atoi(argv[2]);
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	server = gethostbyname(argv[1]);
	if (server == NULL) {
		fprintf(stderr, "ERROR, no such host\n");
		exit(0);
	}
	bzero((char * ) & serv_addr, sizeof(serv_addr));
	serv_addr.sin_family = AF_INET;
	bcopy((char * ) server -> h_addr,(char * ) & serv_addr.sin_addr.s_addr,server -> h_length);
	serv_addr.sin_port = htons(portno);
	if (connect(sockfd, (struct sockaddr * ) & serv_addr, sizeof(serv_addr)) < 0)
		error("ERROR connecting");
	while (1) {
		bzero(buffer, 256);
		printf("Enter message:");
		fgets(buffer, 255, stdin);
		n = write(sockfd, buffer, strlen(buffer));
		if (buffer[0] == '!') break;
		if (n < 0)
		error("ERROR writing to socket");
		bzero(buffer, 256);
		n = read(sockfd, buffer, 255);
		if (buffer[0] == '!') break;
		if (n < 0) error("ERROR reading from socket");
		printf("Message from server:%s\n", buffer);
	}
	close(sockfd);
	return 0;
}
```

Unknown Functions:

- **pthread_exit(NULL):** The "pthread_exit(NULL)" function in C is used to terminate the calling thread and return a status value to the parent thread. The function can be used in a multi-threaded program to exit a thread's execution, rather than terminating the entire program.
- **pthread_join(threads[i], NULL):** The "pthread_join(threads[i], NULL)" function in C is used to wait for the termination of a specified thread, identified by the "threads[i]" parameter. The function blocks the execution of the calling thread until the specified thread has terminated.
- **connect(sockfd, (struct sockaddr *) & serv_addr, sizeof(serv_addr)):** The "connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))" function in C is used to establish a connection to a server with a socket file descriptor "sockfd". The function connects to the server address specified by the "serv_addr" structure, which contains the server's IP address and port number.

4. Give the steps to demonstrate DHCP server.

//In Terminal//

Step 1: $su

Password: Tint@123

If there '#' is present in the end of line it means you are now in SUPER USER. ($ -> #)

Step 2: # ifconfig

(HAVE TO CHECK: NETWORK INTERFACE (enp2s0) & IP Add. <inet:192.168.12.0>). 12, 10, 1: College Server. That's mean you can do work here(system)!

Step 3: apt update

Step 4: apt install isc-dhcp-server –y

Step 5: gedit<space>/etc/default/isc-dhcp-server

//In the Text Editor//

Step 6: <We have to search where 'INTEERFACESv4' present, after that we have to put our system's NETWORK INTERFACE>

INTERFACESv4="enp0s8"(check the interface name by using 'ifconfig')

Save the file!!

Go to, System's Settings > Wired Connection > Network (settings icon) > IPv4 > change AUTOMATIC to MANUAL > change the IP ADD. Value to 192.168.30.1 (this is basically become our DHCP server IP ADD.) and same for Gate Way also (192.168.30.1), change subnet mask 255.255.255.0 & lastly turn off the DNS, save the progress.! When we change these types of settings a small window appears and it ask for ADMIN password & the password is: Tint@156.

Restart Network by turn off and on the Button beside the settings icon of Network.

Check the IP address of the system by using 'ifconfig' putting in terminal.

It must be changed!! (inet: 192.168.30.1)

Step 7: Gedit /etc/dhcp/dhcpd.conf

//Inside the text file//

Task I: We have to find 'option domain-name' (like in line no.10 & 11) and make it comment out by adding # in the front of the line.

Task II: We have to find 'authoritative' and uncomment that by deleting # from the line.

Task III: We have to find '#This is very basic subnet declaration.' And uncomment subnet, range, option routers.

- We have to change the value of the subnet to 192.168.30.0(as we take your server IP ADD. as 192.168.30.1, just changing server IP add.'s host id 1 to 0 <last id of the ADD>) and value of netmask should be 255.255.255.0.

- Change the value of range to 192.168.30.2<space>192.168.30.255 (these are basically, how many clients can be connected through this server)
- Change the value of option routers to 192.168.30.1 (the server IP ADD.)

Save the file!!

//In terminal//

Step 8: # systemctl start isc-dhcp-server

Step 9: # systemctl enable isc-dhcp-server

Step 10:# systemctl status isc-dhcp-server

status must be showing ACTIVATE: Running !! if failed it means error.

5. Give the steps to implement the TELNET Protocol.

Step 1: $ su

Password: Tint@123

Step 2: apt show telnetd

Step 3: apt install telnetd -y

Step 4: systemctl status inetd

for checking the status, by using <sysemctl stop inetd> we can stop the telnet protocol.

Step 5: ufw enable

for enabling the Fire-Wall for TELNET

Step 6: ufw allow 23

Enabling TELNET port

In, Client's system terminal: $ Telnet <SERVER IP Add.>

6. Write the steps to implement FTP Protocol.

Step 1: $ su

Password: Tint@123

Step 2: apt update

Step 3: apt install vsftpd

Step 4: service vsftpd status

Step 5: gedit /etc/vsftpd.conf

refer sample file config.

Step 6: ufw allow from any to any port 20, 21, 10000:10100 proto tcp

Step 7: systemctl restart vsftpd

Step 8: adduser <username>

in, Client's system terminal: $ ftp <server username>

7. Write a program to implement stop and wait.

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

typedef struct packet {
        char data[1024];
}Packet;
typedef struct frame {
```

```
	int frame_kind;
	int sq_no;
	int  ack;
	Packet packet;
}Frame;
void error(const char * msg) {
	perror(msg);
	exit(1);
}
int main(int argc, char * argv[]) {
	int sockfd, newsockfd, portno, f_recv_size;
	socklen_t clilen;
	char buffer[256];
	int frame_id = 0;
	Frame frame_recv;
	Frame frame_send;
	struct sockaddr_in serv_addr, cli_addr;
	int n;
	if (argc < 2) {
		fprintf(stderr, "ERROR, no port provided\n");
		exit(1);
	}
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	bzero((char * ) & serv_addr, sizeof(serv_addr));
	portno = atoi(argv[1]);
	serv_addr.sin_family = AF_INET;
	serv_addr.sin_addr.s_addr = INADDR_ANY;
	serv_addr.sin_port = htons(portno);
	if (bind(sockfd, (struct sockaddr * ) & serv_addr,sizeof(serv_addr)) < 0)
		error("ERROR on binding");
	listen(sockfd, 5);
	clilen = sizeof(cli_addr);
	newsockfd = accept(sockfd,(struct sockaddr * ) & cli_addr, &clilen);
	if (newsockfd < 0)
		error("ERROR on accept");
	while (1) {
		bzero(buffer, 256);
		f_recv_size = read(newsockfd, & frame_recv, sizeof(Frame));
		if (f_recv_size < 0) error("ERROR reading from socket");
		else {
			if (f_recv_size > 0 && frame_recv.frame_kind == 1 && frame_recv.sq_no == frame_id) {
				printf("[+]Frame Received: %s\n", frame_recv.packet.data);
			frame_send.sq_no = 0;
			frame_send.frame_kind = 0;
			frame_send.ack = frame_recv.sq_no + 1;
			n = write(newsockfd, & frame_send, sizeof(frame_send));
			if (n < 0)
				error("ERROR writing to socket");
			else
				printf("[+]Ack Send\n");
			}
			frame_id++;
		}
	}
```

```
	close(newsockfd);
	close(sockfd);
	return 0;
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

typedef struct packet {
	char data[1024];
}Packet;
typedef struct frame {
	int frame_kind;
	int sq_no;
	int  ack;
	Packet packet;
}Frame;
void error(const char * msg) {
	perror(msg);
	exit(0);
}
int main(int argc, char * argv[]) {
	int sockfd, portno, n, f_recv_size;
	struct sockaddr_in serv_addr;
	struct hostent * server;
	char buffer[256];
	int frame_id = 0;
	Frame frame_send;
	Frame frame_recv;
	int ack_recv = 1;
	if (argc < 3) {
		fprintf(stderr, "usage %s hostname port\n", argv[0]);
		exit(0);
	}
	portno = atoi(argv[2]);
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	server = gethostbyname(argv[1]);
	if (server == NULL) {
		fprintf(stderr, "ERROR, no such host\n");
		exit(0);
	}
	bzero((char * ) & serv_addr, sizeof(serv_addr));
	serv_addr.sin_family = AF_INET;
	bcopy((char * ) server -> h_addr,
	(char * ) & serv_addr.sin_addr.s_addr,
	server -> h_length);
	serv_addr.sin_port = htons(portno);
	if (connect(sockfd, (struct sockaddr * ) & serv_addr, sizeof(serv_addr)) < 0)
```

```
            error("ERROR connecting");
        while (1) {
            if (ack_recv == 1) {
                frame_send.sq_no = frame_id;
                frame_send.frame_kind = 1;
                frame_send.ack = 0;
                printf("Please enter the message: ");
                bzero(buffer, 256);
                fgets(buffer, 255, stdin);
                fflush(stdin);
                strcpy(frame_send.packet.data, buffer);
                n = write(sockfd, & frame_send, sizeof(Frame));
                if (n < 0)
                    error("ERROR writing to socket");
                else
                    printf("[+]Frame Send\n");
                bzero(buffer, 256);
                f_recv_size = read(sockfd, & frame_recv, sizeof(frame_recv));
                if (f_recv_size < 0)
                    error("ERROR reading from socket");
                else {
                    if (f_recv_size > 0 && frame_recv.sq_no == 0 && frame_recv.ack == frame_id + 1) {
                        printf("[+]Ack Received\n");
                        ack_recv = 1;
                    } else {
                        printf("[-]Ack Not Received\n");
                        ack_recv = 0;
                        printf("message is %s\n", frame_recv.packet.data);
                    }
                }
            }
            Frame_id++;
        }
        close(sockfd);
        return 0;
}
```

Unknown Functions:

- **fflush(stdin):** The "fflush(stdin)" function in C is used to clear the input buffer of a stream. In this case, the "stdin" stream is being flushed, which clears any pending input from the standard input (keyboard) buffer.

8. Write a program to demonstrate ARQ.

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

typedef struct packet {
        char data[1024];
}Packet;
typedef struct frame {
        int frame_kind;
        int sq_no;
```

```
	int  ack;
	Packet packet;
}Frame;
void error(const char * msg) {
	perror(msg);
	exit(1);
}
int main(int argc, char * argv[]) {
	int sockfd, newsockfd, portno, f_recv_size;
	socklen_t clilen;
	char buffer[256];
	Frame frame_recv;
	Frame frame_send;
	int frame_id = 0;
	struct sockaddr_in serv_addr, cli_addr;
	int n;
	if (argc < 2) {
		fprintf(stderr, "ERROR, no port provided\n");
		exit(1);
	}
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	bzero((char * ) & serv_addr, sizeof(serv_addr));
	portno = atoi(argv[1]);
	serv_addr.sin_family = AF_INET;
	serv_addr.sin_addr.s_addr = INADDR_ANY;
	serv_addr.sin_port = htons(portno);
	if (bind(sockfd, (struct sockaddr * ) & serv_addr, sizeof(serv_addr)) < 0)
		error("ERROR on binding");
	listen(sockfd, 5);
	clilen = sizeof(cli_addr);
	newsockfd = accept(sockfd, (struct sockaddr * ) & cli_addr, & clilen);
	if (newsockfd < 0)
		error("ERROR on accept");
	while (1) {
		bzero(buffer, 256);
		f_recv_size = read(newsockfd, & frame_recv, sizeof(Frame));
		if (f_recv_size < 0)
			error("ERROR reading from socket");
		else {
			if (f_recv_size > 0 && frame_recv.frame_kind == 1 && frame_recv.sq_no == frame_id) {
				printf("[+]Frame Received: %s\n", frame_recv.packet.data);
				frame_send.sq_no = 0;
				frame_send.frame_kind = 0;
				frame_send.ack = frame_recv.sq_no + 1;
				n = write(newsockfd, & frame_send, sizeof(frame_send));
				if (n < 0)
					error("ERROR writing to socket");
				else
					printf("[+]Ack Send\n");
				frame_id = (frame_id + 1) % 2;
			}
		}
	}
	close(newsockfd);
```

```
	return 0;
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

typedef struct packet {
	char data[1024];
}Packet;
typedef struct frame {
	int frame_kind;
	int sq_no;
	int  ack;
	Packet packet;
}Frame;
void error(const char * msg) {
	perror(msg);
	exit(1);
}
int main(int argc, char * argv[]) {
	int sockfd, portno, n, frame_id = 0;
	struct sockaddr_in serv_addr;
	struct hostent * server;
	if (argc < 3) {
		fprintf(stderr, "usage %s hostname port\n", argv[0]);
		exit(0);
	}
	portno = atoi(argv[2]);
	sockfd = socket(AF_INET, SOCK_STREAM, 0);
	if (sockfd < 0)
		error("ERROR opening socket");
	server = gethostbyname(argv[1]);
	if (server == NULL) {
		fprintf(stderr, "ERROR, no such host\n");
		exit(0);
	}
	bzero((char * ) & serv_addr, sizeof(serv_addr));
	serv_addr.sin_family = AF_INET;
	bcopy((char * ) server -> h_addr,(char * ) & serv_addr.sin_addr.s_addr,server -> h_length);
	serv_addr.sin_port = htons(portno);
	if (connect(sockfd, (struct sockaddr * ) & serv_addr, sizeof(serv_addr)) < 0)
		error("ERROR connecting");
	while (1) {
		Frame frame_send;
		Frame frame_recv;
		Packet packet_send;
		Packet packet_recv;
		bzero((char * ) & packet_send, sizeof(packet_send));
		bzero((char * ) & packet_recv, sizeof(packet_recv));
		printf("Enter message: ");
		fgets(packet_send.data, 1023, stdin);
```

```
                frame_send.sq_no = frame_id;
                frame_send.frame_kind = 1;
                frame_send.ack = 0;
                memcpy( & frame_send.packet, & packet_send, sizeof(Packet));
                n = write(sockfd, & frame_send, sizeof(Frame));
                if (n < 0)
                        error("ERROR writing to socket");
                bzero((char * ) & frame_recv, sizeof(frame_recv));
                n = read(sockfd, & frame_recv, sizeof(Frame));
                if (n < 0)
                        error("ERROR reading from socket");
                if (frame_recv.frame_kind == 0 && frame_recv.ack == (frame_id + 1) % 2) {
                        printf("[+]Ack Received\n");
                        frame_id = (frame_id + 1) % 2;
                } else {
                        printf("[-]Invalid Ack\n");
                }
        }
        close(sockfd);
        return 0;
}
```

Unknown Functions:

- **memcpy(& frame_send.packet, & packet_send, sizeof(Packet)):** The "memcpy(&frame_send.packet, &packet_send, sizeof(Packet))" function in C is used to copy the contents of the "packet_send" variable to the "frame_send.packet" variable. The "memcpy" function copies a specified number of bytes from a source memory location to a destination memory location.

9. Write a program to implement Sliding window protocol.

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define WINDOW_SIZE 4
void error(const char * msg) {
        perror(msg);
        exit(1);
}
int main(int argc, char * argv[]) {
        int sockfd, newsockfd, portno;
        socklen_t clilen;
        char buffer[256];
        struct sockaddr_in serv_addr, cli_addr;
        int n, i, j, base = 0;
        int next_seq_num = 0;
        int window_end = base + WINDOW_SIZE;
        int ack[WINDOW_SIZE];
        for (i = 0; i < WINDOW_SIZE; i++)
                ack[i] = 0;
        if (argc < 2) {
                fprintf(stderr, "ERROR, no port provided\n");
```

```
        exit(1);
}
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
        error("ERROR opening socket");
bzero((char * ) & serv_addr, sizeof(serv_addr));
portno = atoi(argv[1]);
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);
if (bind(sockfd, (struct sockaddr * ) & serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR on binding");
listen(sockfd, 5);
clilen = sizeof(cli_addr);
newsockfd = accept(sockfd,(struct sockaddr * ) & cli_addr, &clilen);
if (newsockfd < 0)
        error("ERROR on accept");
while (base < next_seq_num) {
        if (next_seq_num < window_end) {
                bzero(buffer, 256);
                sprintf(buffer, "%d", next_seq_num);
                n = write(newsockfd, buffer, strlen(buffer));
                if (n < 0)
                        error("ERROR writing to socket");
                printf("Sent packet with sequence number %d\n", next_seq_num);
                next_seq_num++;
                window_end++;
        }
        else {
                bzero(buffer, 256);
                n = read(newsockfd, buffer, 255);
                if (n < 0)
                        error("ERROR reading from socket");
                int ack_num = atoi(buffer);
                printf("Received ACK for packet with sequence number %d\n", ack_num);
                for (i = base; i < ack_num; i++) {
                        ack[i % WINDOW_SIZE] = 1;
                }
                while (ack[base % WINDOW_SIZE]) {
                        ack[base % WINDOW_SIZE] = 0;
                        base++;
                        window_end++;
                }
        }
}
bzero(buffer, 256);
sprintf(buffer, "%d", -1);
n = write(newsockfd, buffer, strlen(buffer));
if (n < 0)
        error("ERROR writing to socket");
printf("Sent packet with sequence number -1\n");
close(newsockfd);
```

```
        close(sockfd);
        return 0;
}
```

client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char * msg) {
        perror(msg);
        exit(0);
}
int main(int argc, char * argv[]) {
        int sockfd, portno, n;
        struct sockaddr_in serv_addr;
        struct hostent * server;
        char buffer[256];
        if (argc < 3) {
                fprintf(stderr, "usage %s hostname port\n", argv[0]);
                exit(0);
        }
        portno = atoi(argv[2]);
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0)
                error("ERROR opening socket");
        server = gethostbyname(argv[1]);
        if (server == NULL) {
                fprintf(stderr, "ERROR, no such host\n");
                exit(0);
        }
        bzero((char * ) & serv_addr, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        bcopy((char * ) server -> h_addr,(char * ) & serv_addr.sin_addr.s_addr,server -> h_length);
        serv_addr.sin_port = htons(portno);
        if (connect(sockfd, (struct sockaddr * ) & serv_addr, sizeof(serv_addr)) < 0)
                error("ERROR connecting");
        int i = 0;
        int window_size = 3;
        int seq_no = 0;
        int end_seq_no = strlen(buffer) / window_size;
        while (i <= end_seq_no) {
```

```
            char packet[window_size + 1];
            strncpy(packet, & buffer[i * window_size], window_size);
            packet[window_size] = '\0';
            char seq_no_str[2];
            sprintf(seq_no_str, "%d", seq_no);
            strcat(seq_no_str, packet);
            n = write(sockfd, seq_no_str, strlen(seq_no_str));
            if (n < 0)
                  error("ERROR writing to socket");
            seq_no++;
            i++;
            bzero(packet, window_size + 1);
      }
      n = write(sockfd, "END", 3);
      if (n < 0)
            error("ERROR writing to socket");
      bzero(buffer, 256);
      n = read(sockfd, buffer, 255);
      if (n < 0)
            error("ERROR reading from socket");
      printf("%s\n", buffer);
      close(sockfd);
      return 0;
}
```

Unknown Functions:

- **sprintf(buffer, "%d", next_seq_num):** The "sprintf(buffer, "%d", next_seq_num)" function in C is used to format a string into a buffer. The function converts the integer value "next_seq_num" to a string representation and stores it in the "buffer" variable.
- **atoi(buffer):** The "atoi(buffer)" function in C is used to convert a string representation of an integer in the "buffer" variable to an actual integer value. The "atoi" function parses the string starting from the first character, and stops when it encounters a non-numeric character.
- **strcat(seq_no_str, packet):** The "strcat(seq_no_str, packet)" function in C is used to concatenate two strings: "seq_no_str" and "packet". The function appends the "packet" string to the end of the "seq_no_str" string, effectively joining them together.