1. Write an UDP client server program to send a string ITER from client to server. The server will display the string as well as the client protocol address.

PROGRAM:-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<string.h>
#define MAXLINE 200
#define PORT 33456
void dg_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t client){
int n;
socklen_t len;
struct sockaddr_in *sa;
char buffer[MAXLINE];
memset(buffer,'\0',sizeof(buffer));
len=client;
n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len);
sa=(struct sockaddr_in *)pcliaddr;
printf("Connected client details.\n");
printf("client port number=%d\n",ntohs(sa->sin_port));
printf("client ip details=%s\n",inet_ntoa(sa->sin_addr));
printf("message from client=%s\n",buffer);
}
int main(){
int lisnfd,br; socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr;
len=sizeof(servaddr);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(PORT);
lisnfd=socket(AF_INET, SOCK_DGRAM,0);
if(lisnfd<0){
fprintf(stderr,"create error in socket\n");
return 1;
}
br=bind(lisnfd,(struct sockaddr *)&servaddr, sizeof(servaddr));
if(br==0){
printf("bind success: with return value=%d\n",br);
}
else{
printf("binf unsuccess: with return value=%d\n",br);
printf("Retry different port\n");
exit(2);
}
printf("bind success: with return value=%d\n",br);
printf("binf unsuccess: with return value=%d\n",br); printf("Retry
```

```
different port\n");
exit(2);
dg_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
return 0;
}
Client:
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<string.h>
#define MAXLINE 200
void dg_cli(FILE *FP, int sockfd,const struct sockaddr* pservaddr,
socklen_t servlen){
int n;
char sendline[MAXLINE], recvline[MAXLINE+1];
memset(sendline, '\0', sizeof(sendline));
memset(recvline, '\0', sizeof(recvline));
sendto(sockfd, "ITER",4,0,pservaddr,servlen);
}
int main(int argc,char *argv[]){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
socklen_t len;
len=sizeof(struct sockaddr_in);
if(argc!=3){
fprintf(stderr,"usage <IP> <PORT>\n");
return 1;
}
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(atoi(argv[2]));
sockfd=socket(AF_INET,SOCK_DGRAM,0);
if(sockfd>0){
fprintf(stderr,"socket create success.\n");
}
else{
fprintf(stderr,"create an error.\n");
return 1;
}
printf("Connected server details.\n");
printf("Server port number=%d\n",ntohs(servaddr.sin_port));
printf("Server ip details=%s\n",inet_ntoa(servaddr.sin_addr));
dg_cli(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
return 0;
}
```

2. The client reads a number and sends to the server. The server doubles it and sends back to the client. 2. The client reads a number and sends to the server. The server doubles it and sends back to the client.

PROGRAM:-

Server:-

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define PORT 33456
#define MAXLINE 200
void dg_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t client){
int n;
socklen_t len;
struct sockaddr_in *sa;
char buffer[MAXLINE]; memset(buffer,'\0',sizeof(buffer));
for(;;){
len=client;
n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len);
sa=(struct sockaddr_in*)pcliaddr; printf("connected client details.
     \n");
printf("client port no.=%d\n",ntohs(sa->sin_port)); printf("client IP
details=%s\n",inet_ntoa(sa->sin_addr)); printf("message from
client=%s\n",buffer);
int x=atoi(buffer); x=x*2;
sprintf(buffer, "%d",x); sendto(listenfd,buffer,n,0,pcliaddr,len);
}
}
int main(){
int lisnfd,br;
socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr; len=sizeof(servaddr);
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(PORT); lisnfd=socket(AF_INET,SOCK_DGRAM,0);
if(lisnfd<0){
fprintf(stderr,"create error in socket\n");
return 1;
}
br=bind(lisnfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
if(br==0){
printf("bind success : with return value = %d\n",br);
}
else{
printf("bind unsuccess : with return value = %d\n",br);
printf("retry.  ");
exit(2);
}
dg_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
return 0;
}
```

Client:-

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define MAXLINE 200
void dg_cli(FILE *FP,int sockfd,const struct sockaddr* pservaddr,socklen_t
servlen){
int n;
char sendline[MAXLINE], recvline[MAXLINE+1];
memset(sendline,'\0',sizeof(sendline));
memset(recvline,'\0',sizeof(recvline));
while(1){
printf("enter the number");
fgets(sendline,200,stdin);
sendto(sockfd,sendline,strlen(sendline),0,pservaddr,servlen);
printf("sent data.\n");
n=recvfrom(sockfd,recvline,MAXLINE,0,pservaddr,&servlen);
recvline[n]='\0';
printf("The number doubled = ");
fputs(recvline,stdout); printf("\n");
}
}
int main(int argc,char *argv[]){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
socklen_t len;
len=sizeof(struct sockaddr_in);
if(argc!=3)
{
fprintf(stderr,"usage <IP> <PORT> \n");
return 1;
}
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(atoi(argv[2]));
sockfd=socket(AF_INET,SOCK_DGRAM,0);
if(sockfd>0){
 fprintf(stderr,"success in socket creation \n");
}
else{
fprintf(stderr ,"create an error\n");
return 1 ;
}
printf("connected server details.\n");
printf("server port number %d\n",ntohs(servaddr.sin_port));
dg_cli(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
return 0;
}
```

3. The client reads a line of words separated by white space and sends to the server. The server trans forms the line of words in which the words appear in the reverse order and sends back to the client. The client displays original line and the received line from the server. For example Alice likes Bob transforms to Bob likes Alice. Implement a function for reversing the words in a string s.

PROGRAM:-

Server:-

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define PORT 33456
#define MAXLINE 200
void reverse(char* begin, char* end){
char temp;
while (begin < end) {
temp = *begin;
*begin++ = *end;
*end-- = temp;
}
}
void reverseWords(char* s)
{
char* word_begin = s;
while (*temp) {
temp++;
if (*temp == '\0') {
reverse(word_begin, temp - 1);
}
else if (*temp == ' ') {
reverse(word_begin, temp - 1);
word_begin = temp + 1;
}
}
reverse(s, temp - 1);
}
void dg_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t client){
int n;
socklen_t len;
struct sockaddr_in *sa;
char buffer[MAXLINE]; memset(buffer,'\0',sizeof(buffer));
for(;;){
len=client; n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len);
sa=(struct sockaddr_in*)pcliaddr;
printf("connected client details.     \n");
printf("client port no.=%d\n",ntohs(sa->sin_port)); printf("client IP
details=%s\n",inet_ntoa(sa->sin_addr)); printf("message from
client=%s\n",buffer); reverseWords(buffer);
```

```
sendto(listenfd,buffer,n,0,pcliaddr,len);
}
}
int main(){
int lisnfd,br; socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr; len=sizeof(servaddr);
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(PORT); lisnfd=socket(AF_INET,SOCK_DGRAM,0);
if(lisnfd<0){
fprintf(stderr,"create error in socket\n"); return 1;
}
br=bind(lisnfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
if(br==0){
 printf("bind success : with return value = %d\n",br);
}
else{
printf("bind unsuccess : with return value = %d\n",br);
printf("retry.  ");
exit(2);
}
dg_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
return 0;
}

Client:-
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define MAXLINE 200
void dg_cli(FILE *FP,int sockfd,const struct sockaddr* pservaddr,socklen_t
servlen){
int n;
char sendline[MAXLINE], recvline[MAXLINE+1];
memset(sendline,'\0',sizeof(sendline));
memset(recvline,'\0',sizeof(recvline));
while(1){
printf("\nenter the string.");
fgets(sendline,200,stdin);
sendto(sockfd,sendline,strlen(sendline),0,pservaddr,servlen);
printf("sent data.\n");
n=recvfrom(sockfd,recvline,MAXLINE,0,pservaddr,&servlen);
recvline[n]='\0';
printf("Reverse String : \n");
fputs(recvline,stdout);
}
}
int main(int argc,char *argv[]){
int sockfd;
```

```
struct sockaddr_in servaddr,cliaddr;
socklen_t len;
len=sizeof(struct sockaddr_in);
if(argc!=3){
fprintf(stderr,"usage <IP> <PORT> \n");
return 1;
}
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(atoi(argv[2]));
sockfd=socket(AF_INET,SOCK_DGRAM,0);
if(sockfd>0){
fprintf(stderr,"success in socket creation \n");
}
else
fprintf(stderr ,"create an error\n");
return 1 ;
}
printf("connected server details.\n");
printf("server port number %d\n",ntohs(servaddr.sin_port));
dg_cli(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
return 0;
}
```

4. The client writes a datagram of length 0 and sends to the server. Verify that it is acceptable and display the same in server side.

PROGRAM:-

Server:-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<string.h>
#define MAXLINE 200
#define PORT 33456
void dg_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t client){
int n; socklen_t len;
struct sockaddr_in *sa; char buffer[MAXLINE];
memset(buffer,'\0',sizeof(buffer)); len=client;
n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len); sa=(struct
sockaddr_in *)pcliaddr;
printf("Connected client details.     \n");
printf("client port number=%d\n",ntohs(sa->sin_port));
printf("client ip details=%s\n",inet_ntoa(sa->sin_addr));
printf("message from client=%s\n",buffer);
}
int main(){
int lisnfd,br; socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr; len=sizeof(servaddr);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
```

```
servaddr.sin_port=htons(PORT); lisnfd=socket(AF_INET, SOCK_DGRAM,0);
if(lisnfd<0){
fprintf(stderr,"create error in socket\n");
return 1;
}
br=bind(lisnfd,(struct sockaddr *)&servaddr, sizeof(servaddr));
if(br==0){
printf("bind success: with return value=%d\n",br);
}
else{
printf("binf unsuccess: with return value=%d\n",br); printf("Retry
different port\n");
exit(2);
}
dg_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
return 0;
}

Client:
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<string.h>
#define MAXLINE 200
void dg_cli(FILE *FP, int sockfd,const struct sockaddr* pservaddr,
socklen_t servlen){
int n;
char sendline[MAXLINE], recvline[MAXLINE+1];
memset(sendline, '\0', sizeof(sendline));
memset(recvline, '\0', sizeof(recvline));
sendto(sockfd, "",4,0,pservaddr,servlen);
}
int main(int argc,char *argv[]){
int sockfd;
struct sockaddr_in servaddr,cliaddr; socklen_t len;
len=sizeof(struct sockaddr_in);
if(argc!=3){
fprintf(stderr,"usage <IP> <PORT>\n");
return 1;
}
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(atoi(argv[2]));
sockfd=socket(AF_INET,SOCK_DGRAM,0);
if(sockfd>0){
fprintf(stderr,"socket create success\n");
}
else
{
```

```
fprintf(stderr,"create an error  \n");
return 1;
}
printf("Connected server details\n");
printf("Server port number=%d\n",ntohs(servaddr.sin_port));
printf("Server ip details=%s\n",inet_ntoa(servaddr.sin_addr));
dg_cli(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
return 0;
}
```

5. Let an UDP client sends a number to UDP server and the server finds the sum of the digits of the received number. The server sends the sum to the client. You have to modify the 4th parameter, from, argument to recvfrom is a null pointer and the corresponding 5th parameter, addrlen, also to be a null pointer to indicate that the server is not interested to know the protocol address of who send the data.

PROGRAM:-

Server:-

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define PORT 33456
#define MAXLINE 200
int getSum(int n)
{
int sum;
for (sum = 0; n > 0; sum += n % 10, n /= 10);
return sum;
}
void dg_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t client)
{
int n;
socklen_t len;
struct sockaddr_in *sa;
char buffer[MAXLINE]; memset(buffer,'\0',sizeof(buffer));
for(;;){
len=client;
n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len);
sa=(struct sockaddr_in*)pcliaddr; printf("connected client details.\n");
printf("client port no.=%d\n",ntohs(sa->sin_port)); printf("client IP
details=%s\n",inet_ntoa(sa->sin_addr));
printf("message from client=%s\n",buffer);
int x=atoi(buffer); int sum=getSum(x);
sprintf(buffer, "%d",sum); sendto(listenfd,buffer,n,0,pcliaddr,len);
}
}
int main(){
int lisnfd,br;
socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr; len=sizeof(servaddr);
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
```

```
servaddr.sin_port=htons(PORT); lisnfd=socket(AF_INET,SOCK_DGRAM,0);
if(lisnfd<0){
fprintf(stderr,"create error in socket\n"); return 1;
}
br=bind(lisnfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
if(br==0){
 printf("bind success : with return value = %d\n",br);
}
else
{
printf("bind unsuccess : with return value = %d\n",br); printf("retry. ");
exit(2);
}
dg_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
return 0;
}

Client:
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define MAXLINE 200
void dg_cli(FILE *FP,int sockfd,const struct sockaddr* pservaddr,socklen_t
servlen)
{
      int n;
      char sendline[MAXLINE], recvline[MAXLINE+1];
      memset(sendline,'\0',sizeof(sendline));
      memset(recvline,'\0',sizeof(recvline));
      while(1)
      {
      printf("enter the number");
      fgets(sendline,200,stdin);
      sendto(sockfd,sendline,strlen(sendline),0,pservaddr,servlen);
      printf("sent data.\n");
      n=recvfrom(sockfd,recvline,MAXLINE,0,NULL,NULL);
      recvline[n]='\0';
      printf("The sum of digits of the number = ");
      fputs(recvline,stdout);
      printf("\n");
      }
}
int main(int argc,char *argv[])
{
      int sockfd;
      struct sockaddr_in servaddr,cliaddr;
      socklen_t len;
      len=sizeof(struct sockaddr_in);
```

```
      if(argc!=3)
      {
            fprintf(stderr,"usage <IP> <PORT> \n");
            return 1;
      }
      servaddr.sin_family=AF_INET;
      servaddr.sin_addr.s_addr=inet_addr(argv[1]);
      servaddr.sin_port=htons(atoi(argv[2]));
      sockfd=socket(AF_INET,SOCK_DGRAM,0);
      if(sockfd>0)
      {
fprintf(stderr,"success in socket creation \n");
      }
      else {
            fprintf(stderr ,"create an error \n");return 1 ;
      }
      printf("connected server details.\n");
      printf("server port number %d\n",ntohs(servaddr.sin_port));
      dg_cli(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
      return 0;
}
```

6. Design a TCP client server for sending and receiving a string application using recvfrom and sendto instead of read and write. State the reason, why normally it is not required to do this in TCP client server application.

PROGRAM:-

Server:-

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
int main(int argc, char **argv){
int listenfd,connfd,len;
struct sockaddr_in servaddr,clientaddr;
char buff[1024];
time_t ticks;
len=sizeof(struct sockaddr_in);
listenfd=socket(AF_INET, SOCK_STREAM,0);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(0);
bind(listenfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
getsockname(listenfd,(struct sockaddr *)&servaddr,&len);
printf("after bind ephermeral
port=%d\n",(int)ntohs(servaddr.sin_port));
listen(listenfd,5);
connfd=accept(listenfd,(struct sockaddr *)&clientaddr, &len);
ticks=time(NULL);
snprintf(buff,sizeof(buff),"%s\r\n",ctime(&ticks));
sendto(connfd,buff,strlen(buff),0,NULL,NULL);
sendto(connfd,"ITER",4,0,NULL,NULL);
close(connfd);
```

```
    }
Client:-
    #include<stdio.h>
    #include<sys/socket.h>
    #include<sys/types.h>
    #include<netinet/in.h>
    #include<arpa/inet.h>
    #include<stdlib.h>
    #include<string.h>
    #include<time.h>
    int main(int argc, char *argv[]){
    int sockfd,n,conn,len;
    char recvline[1024];
    struct sockaddr_in servaddr;
    len=sizeof(struct sockaddr_in);
    sockfd=socket(AF_INET, SOCK_STREAM, 0);
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr(argv[1]);
    servaddr.sin_port=htons(atoi(argv[2]));
    connect(sockfd,(struct sockaddr *)&servaddr,sizeof(servaddr));
    n=recvfrom(sockfd,recvline,1024,0,NULL,NULL);
    printf("%d\n",n); recvline[n]=0; printf("%s",recvline);
    close(sockfd);
    }
```

7. The client reads 10 numbers and sends them to the server one by one. The server displays them one after another.

PROGRAM:-

```
Server:-
    #include<stdio.h>
    #include<sys/socket.h>
    #include<sys/types.h>
    #include<sys/socket.h>
    #include<netinet/in.h>
    #include<arpa/inet.h>
    #include<stdlib.h>
    #include<string.h>
    #include<time.h>
    #define PORT 33456
    #define MAXLINE 200
    void dg_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t
    client){
    int n; socklen_t len;
    struct sockaddr_in *sa;
    char buffer[MAXLINE]; memset(buffer,'\0',sizeof(buffer));
    for(;;){
    len=client;
    n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len);
    sa=(struct sockaddr_in*)pcliaddr;
    printf("message from client=%s\n",buffer);
    }
    }
```

```
int main(){
int lisnfd,br;
socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr; len=sizeof(servaddr);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(PORT);
lisnfd=socket(AF_INET,SOCK_DGRAM,0);
if(lisnfd<0){
fprintf(stderr,"create error in socket\n"); return 1;
}
br=bind(lisnfd,(struct
sockaddr*)&servaddr,sizeof(servaddr));if(br==0){
 printf("bind success : with return value = %d\n",br);
}
else{
printf("bind unsuccess : with return value = %d\n",br);
printf("retry.");
exit(2);
}
dg_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
return 0;
}
```

Client:-

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define MAXLINE 200
void dg_cli(FILE *FP,int sockfd,const struct sockaddr*
pservaddr,socklen_t servlen){
int n;
char sendline[MAXLINE], recvline[MAXLINE+1];
memset(sendline,'\0',sizeof(sendline));
memset(recvline,'\0',sizeof(recvline));
for(int i=0;i<10;i++){
printf("enter the number ");
fgets(sendline,200,stdin);
sendto(sockfd,sendline,strlen(sendline),0,pservaddr,servlen);
printf("Data sent successfully\n");
}
}
int main(int argc,char *argv[]){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
socklen_t len;
len=sizeof(struct sockaddr_in);
if(argc!=3){
```

```
fprintf(stderr,"usage <IP> <PORT> \n");return 1;
}
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(atoi(argv[2]));
sockfd=socket(AF_INET,SOCK_DGRAM,0);
if(sockfd>0){
fprintf(stderr,"success in socket creation \n");
}
else{
fprintf(stderr ,"create an error\n");return 1 ;
}
printf("connected server details.\n");
printf("server port number %d\n",ntohs(servaddr.sin_port));
dg_cli(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
return 0;
}
```

8. Design an UDP server program and two UDP client programs. The first client will send a string in small case to the server. The server will display the string along with the clients who sends the data. Now the server will send the upper case of the string to the second client and the second client will display the string forwarded from the server.

PROGRAM:-

Server:-

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<string.h>
#define MAXLINE 200
#define PORT 33456
void dg_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t client){
int n;
socklen_t len;
struct sockaddr_in *sa; char buffer[MAXLINE];
memset(buffer,'\0',sizeof(buffer));
len=client;
n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len); sa=(struct sockaddr_in
*)pcliaddr;
printf("Connected client details.\n");
printf("client port number=%d\n",ntohs(sa->sin_port)); printf("client ip
details=%s\n",inet_ntoa(sa->sin_addr));
printf("message from client=%s\n",buffer);
}
int main(){
int lisnfd,br; socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr; len=sizeof(servaddr);
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY); servaddr.sin_port=htons(PORT);
lisnfd=socket(AF_INET, SOCK_DGRAM,0);
if(lisnfd<0){
```

```
fprintf(stderr,"create error in socket\n"); return 1;
}
br=bind(lisnfd,(struct sockaddr *)&servaddr, sizeof(servaddr));
if(br==0){
printf("bind success: with return value=%d\n",br);
}
else{
printf("binf unsuccess: with return value=%d\n",br); printf("Retry
different port\n");exit(2);
}
dg_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr)); return 0;
}
 Client:-
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<netdb.h>
#include<string.h>
#define MAXLINE 200
void dg_cli(FILE *FP, int sockfd,const struct sockaddr* pservaddr,
socklen_t servlen){
int n;
char sendline[MAXLINE], recvline[MAXLINE+1]; memset(sendline, '\0',
sizeof(sendline)); memset(recvline, '\0', sizeof(recvline)); sendto(sockfd,
"COVID-19",10,0,pservaddr,servlen);
}
int main(int argc,char *argv[]){
int sockfd;
struct sockaddr_in servaddr,cliaddr; socklen_t len;
len=sizeof(struct sockaddr_in);
if(argc!=3){
fprintf(stderr,"usage <IP> <PORT>\n"); return 1;
}
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(atoi(argv[2]));
sockfd=socket(AF_INET,SOCK_DGRAM,0);
if(sockfd>0)
{
fprintf(stderr,"socket create success\n");
}
else{
fprintf(stderr,"create an error.\ n");return 1;
}
printf("Connected server details.\n");
printf("Server port number=%d\n",ntohs(servaddr.sin_port));
printf("Server ip details=%s\n",inet_ntoa(servaddr.sin_addr));
dg_cli(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));return 0;
}
```

14. [UDP Echo Server:] Design an UDP client/server program for an echo server that performs the following

steps: [Refer chapter 8.3 of your text book]

i) The client reads a line of text from its standard input and write the line to the server.
ii) The server reads the line from its network input and echoes the line back to the client.
iii) The client reads the echoed line and prints its on its standard output.
iv) The server will keep a track of the client number and the protocol address that is connected to the server.
v) The UDP echo server will use a user-defined function, datagram echo(), to echo lines on a datagram socket.
vi) The client will also call a user-defined function, datagram client(), to send line of text to the server.
vii) In the client side, the function datagram client() must meet the given four steps such as (1) read a line from the standard input using fgets, (2) send the line to the server using sendto, (3) read back the server's echo using recvfrom and (4) print the echoed line to the standard output using fputs respectively in a loop till you type EOF character (CTRL+D), which terminate the clien

PROGRAM:-

Server:-

```
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define PORT 33456
#define MAXLINE 200
void datagram_echo(int listenfd,struct sockaddr* pcliaddr,socklen_t
client){
int n; socklen_t len;
struct sockaddr_in *sa;
char buffer[MAXLINE]; memset(buffer,'\0',sizeof(buffer));
for(;;){
len=client; n=recvfrom(listenfd,buffer,MAXLINE,0,pcliaddr,&len);
sa=(struct sockaddr_in*)pcliaddr; printf("connected client details.
\n");
printf("client port no.=%d\n",ntohs(sa->sin_port)); printf("client IP
details=%s\n",inet_ntoa(sa->sin_addr)); printf("message from
client=%s\n",buffer);
sendto(listenfd,buffer,n,0,pcliaddr,len);
}}
int main(){
int lisnfd,br; socklen_t clilen,len;
struct sockaddr_in servaddr,cliaddr; len=sizeof(servaddr);
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(PORT);
lisnfd=socket(AF_INET,SOCK_DGRAM,0);
if(lisnfd<0){
fprintf(stderr,"create error in socket\n");
return 1;
}
br=bind(lisnfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
if(br==0){
printf("bind success : with return value = %d\n",br);
}
else{
printf("bind unsuccess : with return value = %d\n",br); printf("retry.");
```

```
exit(2);
}
datagram_echo(lisnfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
return 0;
}
Client:-
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#define MAXLINE 200
void datagram_client(FILE *FP,int sockfd,const struct sockaddr*
pservaddr,socklen_t servlen){
int n;
char sendline[MAXLINE], recvline[MAXLINE+1];
memset(sendline,'\0',sizeof(sendline));
memset(recvline,'\0',sizeof(recvline));
while(1){
printf("enter the message");
fgets(sendline,200,stdin);
sendto(sockfd,sendline,strlen(sendline),0,pservaddr,servlen);
printf("sent data\n");
n=recvfrom(sockfd,recvline,MAXLINE,0,pservaddr,&servlen);
recvline[n]='\0';
printf("Message from server : "); fputs(recvline,stdout);
}
}
int main(int argc,char *argv[]){
int sockfd;
struct sockaddr_in servaddr,cliaddr;
socklen_t len;
len=sizeof(struct sockaddr_in);
if(argc!=3){
fprintf(stderr,"usage <IP> <PORT> \n"); return 1;
}
servaddr.sin_family=AF_INET; servaddr.sin_addr.s_addr=inet_addr(argv[1]);
servaddr.sin_port=htons(atoi(argv[2]));
sockfd=socket(AF_INET,SOCK_DGRAM,0);
if(sockfd>0){
fprintf(stderr,"success in socket creation \n");
}
else{
fprintf(stderr ,"create an error \n");return 1 ;
}
printf("connected server details.     \n");
printf("server port number %d\n",ntohs(servaddr.sin_port));
datagram_client(stdin,sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
return 0;
}
```