NPS LAB PROGRAMS

1. Implement a client and server communication using sockets programming.

Server.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
#include <arpa/inet.h>

int main()
{
	int cont,create_socket,new_socket,addrlen,fd;
	int bufsize = 1024;
	char *buffer = malloc(bufsize);
	char fname[256];
	struct sockaddr_in address;

	if ((create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0)
		printf("The socket was created\n");

	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons(15000);

	if (bind(create_socket,(struct sockaddr *)&address,sizeof(address)) == 0)
		printf("Binding Socket\n");

	listen(create_socket,3);
	addrlen = sizeof(struct sockaddr_in);
	new_socket = accept(create_socket,(struct sockaddr *)&address,&addrlen);

	if (new_socket > 0)
		printf("The Client %s is Connected...\n ", inet_ntoa(address.sin_addr));

	recv(new_socket,fname, 255,0);
	printf("A request for filename %s Received..\n", fname);

	if ((fd=open(fname, O_RDONLY))<0)
```

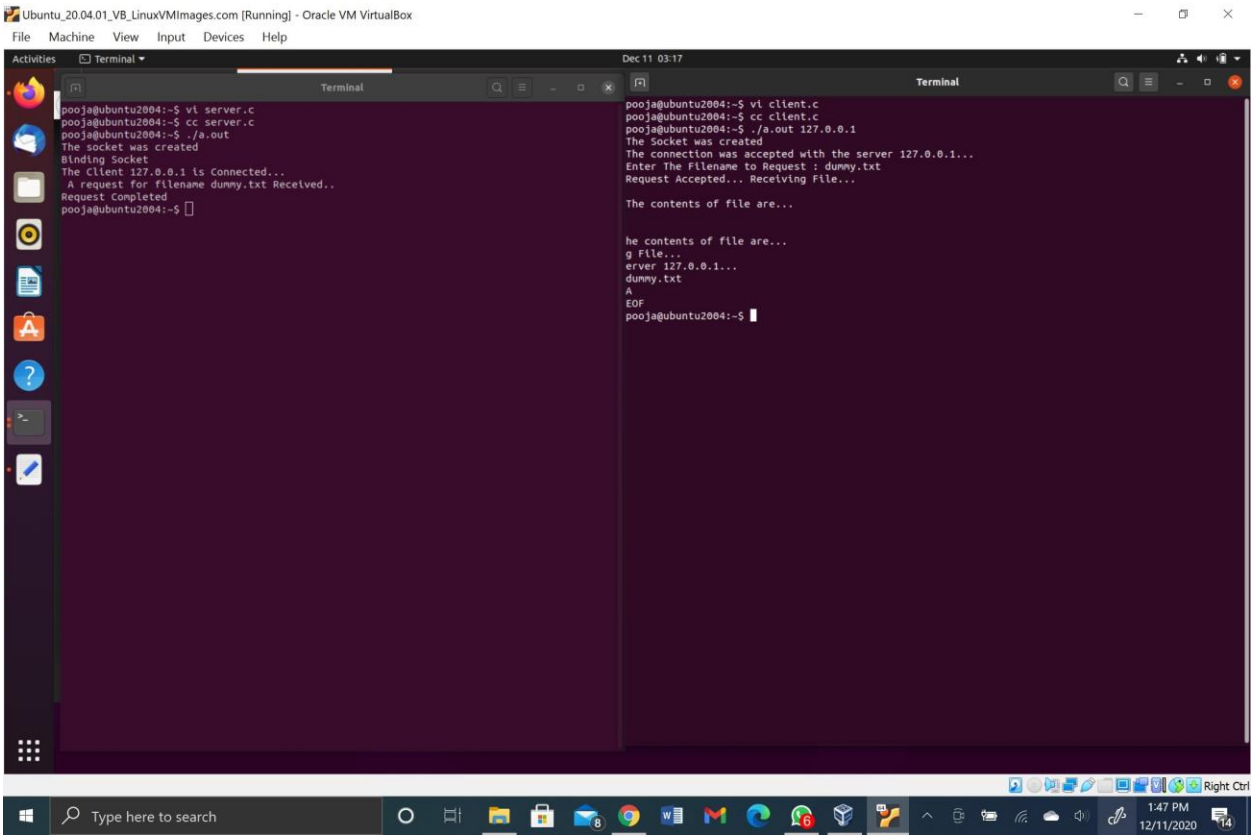
```
        {
                perror("File Open Failed"); exit(0);}
                while((cont=read(fd, buffer, bufsize))>0) {
                send(new_socket,buffer,cont,0);
        }

        printf("Request Completed\n");

        close(new_socket);
        return close(create_socket);
}
```

Client.c

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

int main(int argc,char *argv[ ])
{
        int create_socket;
        int bufsize = 1024;
        char *buffer = malloc(bufsize);
        char fname[256];
        struct sockaddr_in address;

        if ((create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0)
                printf("The Socket was created\n");

        address.sin_family = AF_INET;
        address.sin_port = htons(15000);
        inet_pton(AF_INET,argv[1],&address.sin_addr);

        if (connect(create_socket,(struct sockaddr *) &address, sizeof(address)) == 0)
                printf("The connection was accepted with the server %s...\n",argv[1]);

        printf("Enter The Filename to Request : ");
        scanf("%s",fname);
        send(create_socket, fname, sizeof(fname), 0);
        printf("Request Accepted... Receiving File...\n\n");
```

```
        printf("The contents of file are...\n\n");

        uint32_t cont;
        while((cont=recv(create_socket, buffer, bufsize, 0))>0) {
                write(1, buffer, cont);
        }

        printf("\nEOF\n");
        return close(create_socket);
}
```

Output



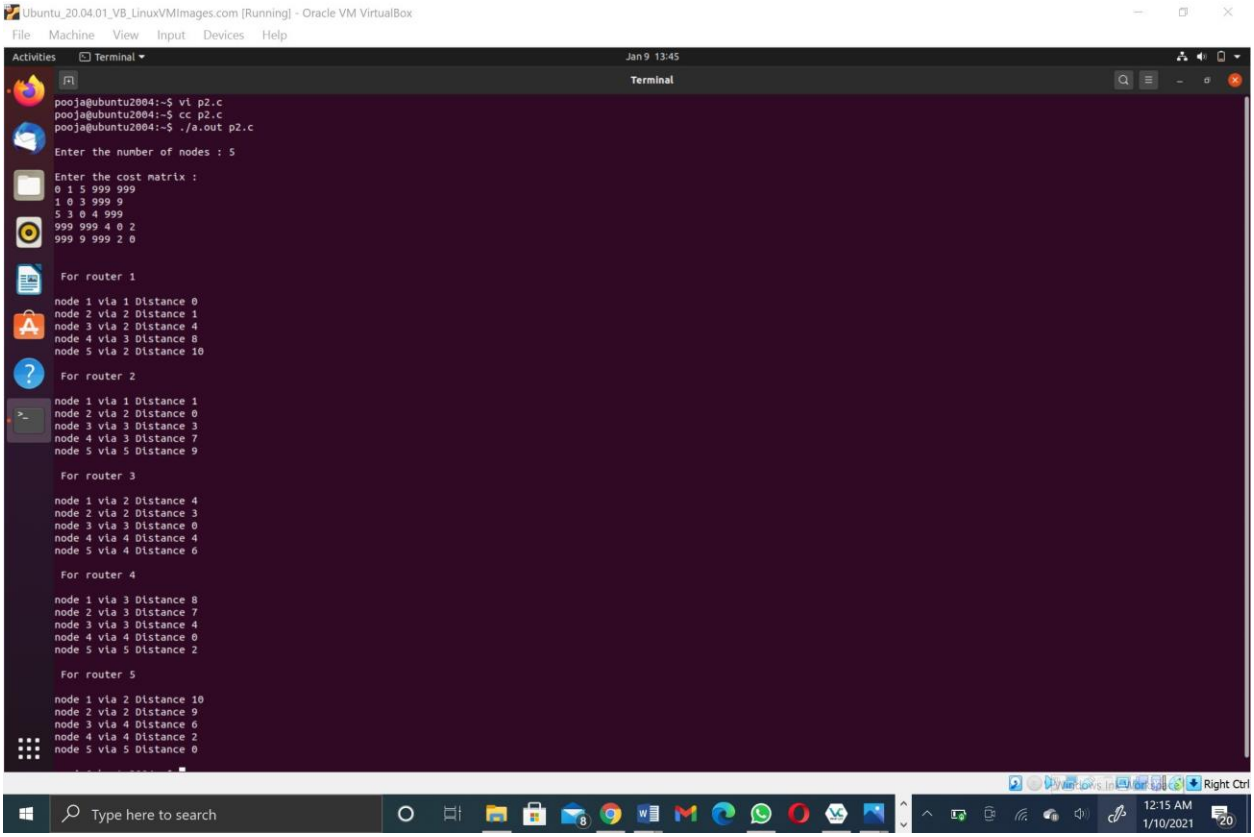
2. Write a program to implement distance vector routing protocol for a simple topology of routers.

```
#include<stdio.h>
struct node
{
   unsigned dist[20];
```

```
    unsigned from[20];
}rt[10];
int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes);//Enter the nodes
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
        {
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix
            rt[i].from[j]=j;
        }
    }
        do
        {
            count=0;
            for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct
distance from the node i to k using the cost matrix
            //and add the distance from k to node j
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {//We calculate the minimum distance
                    rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                    rt[i].from[j]=k;
                    count++;
                }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
            for(j=0;j<nodes;j++)
            {
                printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
            }
        }
    printf("\n\n");
```

}

Output:



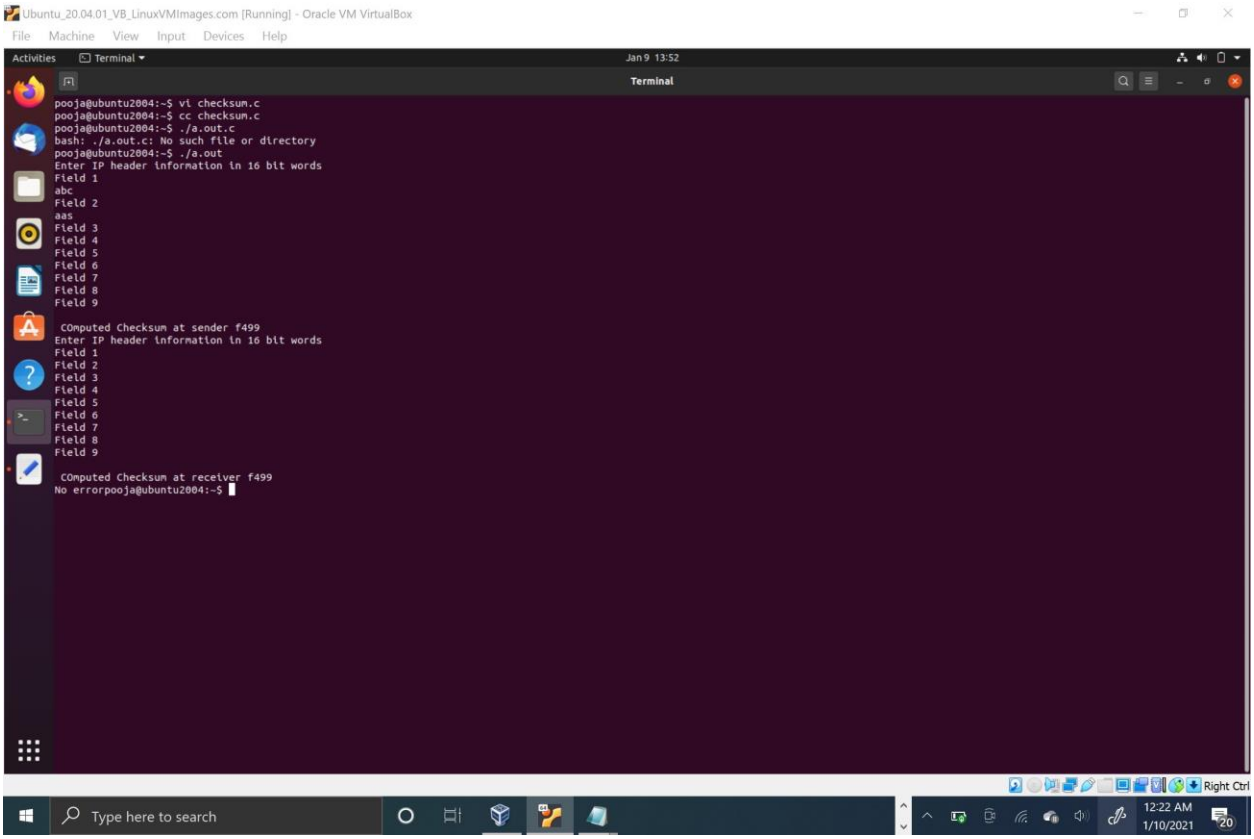
3. Write a program to implement error detection and correction concept using Checksum and Hamming code

Checksum

```
#include<stdio.h>
unsigned fields[10];
unsigned short checksum()
{
int i;
int sum=0;
printf("Enter IP header information in 16 bit words\n");
for(i=0;i<9;i++)
{
printf("Field %d\n",i+1);
scanf("%x",&fields[i]);
sum=sum+(unsigned short)fields[i];
while (sum>>16)
sum = (sum & 0xFFFF) + (sum >> 16);
}
```

```
sum=~sum;
return (unsigned short)sum;
}
int main()
{
unsigned short result1, result2;
//Sender
result1=checksum();
printf("\n COmputed Checksum at sender %x\n", result1);
//Receiver
result2=checksum();
printf("\n COmputed Checksum at receiver %x\n", result2);
if(result1==result2)
printf("No error");
else
printf("Error in data received");
}
```

Output:



Hamming code

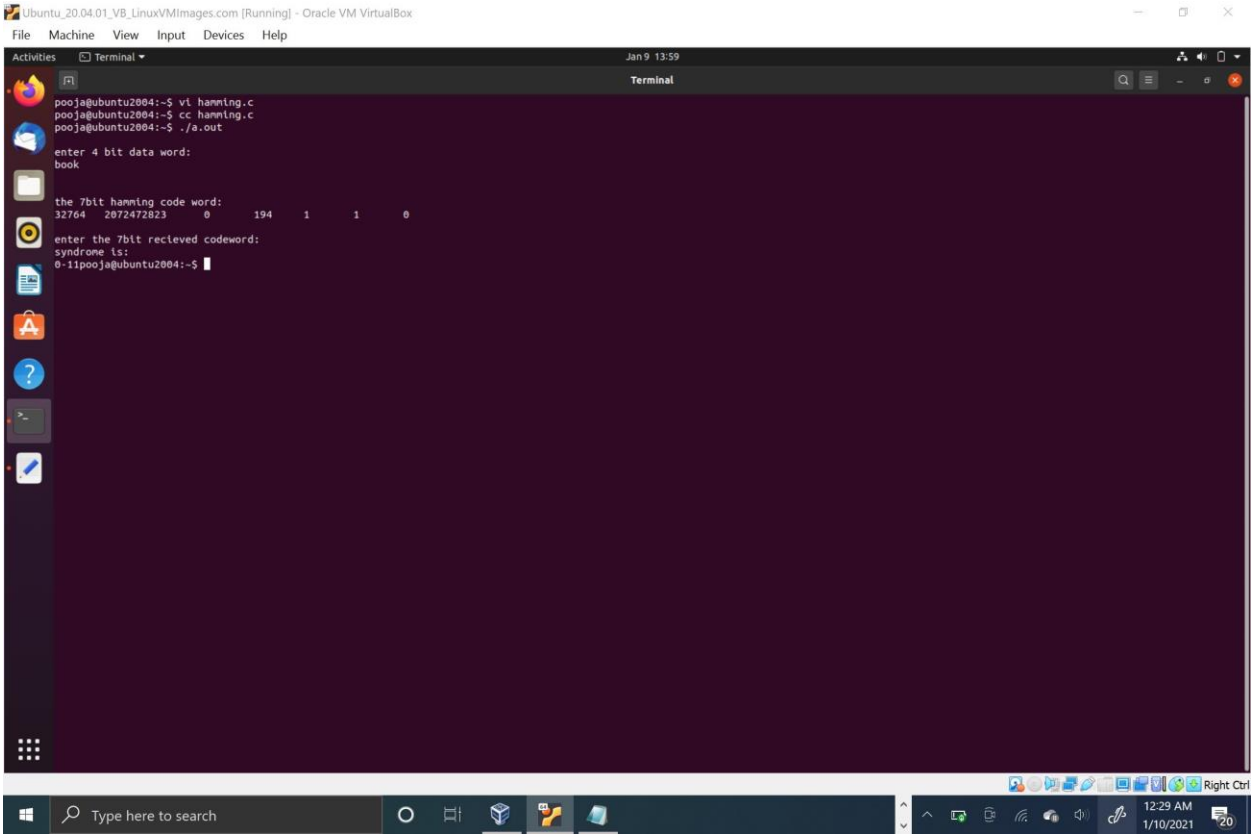
```
#include <stdlib.h>
```

```
#include<stdio.h>
int main()
{
    int a[4],b[4],r[3],s[3],i,q[3],c[7];
    printf("\nenter 4 bit data word:\n");
    for(i=3;i>=0;i--)
    {
    scanf("%d",&a[i]);
    }
    r[0]=(a[3]+a[1]+a[0])%2;
    r[1]=(a[0]+a[2]+a[3])%2;
    r[2]=(a[1]+a[2]+a[3])%2;
    printf("\n\nthe 7bit hamming code word: \n");
    for(i=3;i>=0;i--)
    {
    printf("%d\t",a[i]);
    }
    for(i=2;i>=0;i--)
    {
    printf("%d\t",r[i]);
    }
    printf("\n");
    printf("\nenter the 7bit recieved codeword: ");
    for(i=7;i>0;i--)
    scanf ("%d",&c[i]);
    b[3]=c[7];b[2]=c[6];b[1]=c[5];b[0]=c[4];
    r[2]=c[3];r[1]=c[2];r[0]=c[1];
    //calculating syndrome bits
    s[0]=(b[0]+b[1]+b[3]+r[0])%2;
    s[1]=(b[0]+b[2]+b[3]+r[1])%2;
    s[2]=(b[1]+b[2]+b[3]+r[2])%2;
    printf("\nsyndrome is: \n");
    for(i=2;i>=0;i--)
    {
    printf("%d",s[i]);
    }
    if((s[2]==0) && (s[1]==0) && (s[0]==0))
    printf("\n RECIEVED WORD IS ERROR FREE\n");
    if((s[2]==1)&&(s[1]==1)&&(s[0]==1))
    {
    printf("\nError in received codeword, position- 7th bit from right\n");
    if(c[7]==0)
    c[7]=1;
```

```
else
c[7]=0;
printf("\n Corrected codeword is\n");
for(i=7;i>0;i--)
printf("%d \t", c[i]);
}
if((s[2]==1)&&(s[1]==1)&&(s[0]==0))
{
printf("\nError in received codeword, Position- 6th bit from right\n");
if(c[6]==0)
c[6]=1;
else
c[6]=0;
printf("\n Corrected codeword is\n");
for(i=7;i>0;i--)
printf("%d \t", c[i]);
}
if((s[2]==1)&&(s[1]==0)&&(s[0]==1))
{
printf("\nError in received codeword, Position- 5th bit from right\n");
if(c[5]==0)
c[5]=1;
else
c[5]=0;
printf("\n Corrected codeword is\n");
for(i=7;i>0;i--)
printf("%d \t", c[i]);
}
if((s[2]==1)&&(s[1]==0)&&(s[0]==0))
{
printf("\nError in received codeword, Position- 4th bit from right\n");
if(c[4]==0)
c[4]=1;
else
c[4]=0;
printf("\n Corrected codeword is\n");
for(i=7;i>0;i--)
printf("%d \t", c[i]);
}
if((s[2]==0)&&(s[1]==1)&&(s[0]==1))
{
   printf("\nError in received codeword, Position- 3rd bit from right\n");
   if(c[3]==0)
```

```
            c[3]=1;
        else
            c[3]=0;
        printf("\n Corrected codeword is\n");
        for(i=7;i>0;i--)
            printf("%d \t", c[i]);
    }
    if((s[2]==0)&&(s[1]==1)&&(s[0]==0))
    {
        printf("\nError in received codeword, Position- 2nd bit from right\n");
        if(c[2]==0)
            c[2]=1;
        else
            c[2]=0;
        printf("\n Corrected codeword is\n");
        for(i=7;i>0;i--)
            printf("%d \t", c[i]);
    }
    if((s[2]==0)&&(s[1]==0)&&(s[0]==1))
    {
        printf("\nError in received codeword, Position- 1st bit from right\n");
        if(c[1]==0)
            c[1] =1;
        else
            c[1]=0;
        printf("\n Corrected codeword is\n");
        for(i=7;i>0;i--)
            printf("%d \t", c[i]);
    }
    return(1);
}//End of Hamming code program*/
```

Output:



4. Implementation of a simple multicast routing mechanism.

Sender.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include<stdlib.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
int main(int argc, char *argv[])
{
struct sockaddr_in addr;
int fd, cnt;
struct ip_mreq mreq;
char *message="RVCE-CSE";
/* create what looks like an ordinary UDP socket */
if ((fd=socket(AF_INET,SOCK_DGRAM,0)) < 0) {
perror("socket");
```

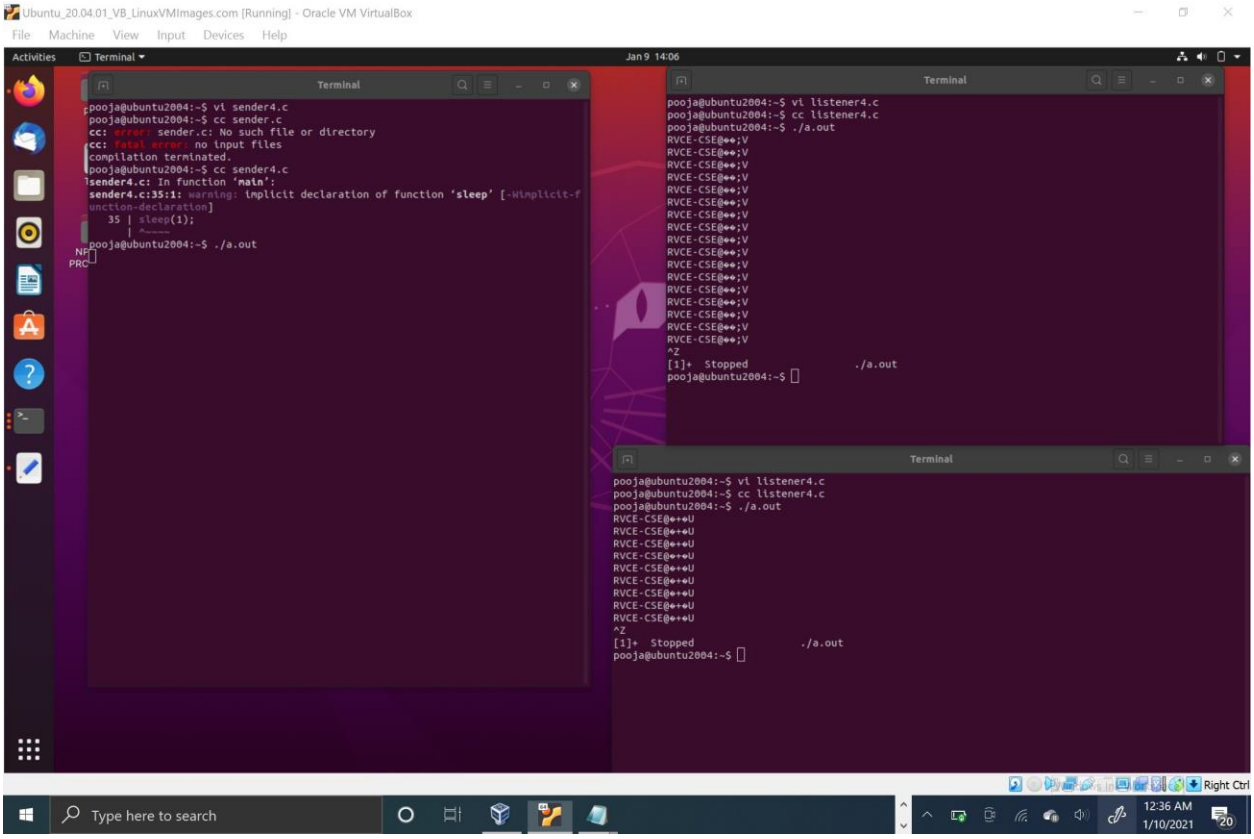
```
exit(1);
}
/* set up destination address */
memset(&addr,0,sizeof(addr));
addr.sin_family=AF_INET;
addr.sin_addr.s_addr=inet_addr(HELLO_GROUP);
addr.sin_port=htons(HELLO_PORT);
/* now just sendto() our destination! */
while (1) {
if (sendto(fd,message,sizeof(message),0,(struct sockaddr *)
&addr,
sizeof(addr)) < 0) {
perror("sendto");
exit(1);
}
sleep(1);
}
return 0;
}
```

Listener.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include<stdlib.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
#define MSGBUFSIZE 25
int main(int argc, char *argv[])
{
struct sockaddr_in addr;
int fd, nbytes,addrlen;
struct ip_mreq mreq;
char msgbuf[MSGBUFSIZE];
u_int yes=1; /*** MODIFICATION TO ORIGINAL */
/* create what looks like an ordinary UDP socket */
if ((fd=socket(AF_INET,SOCK_DGRAM,0)) < 0) {
perror("socket");
exit(1);
```

```
}
/**** MODIFICATION TO ORIGINAL */
/* allow multiple sockets to use the same PORT number */
if (setsockopt(fd,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(yes)) < 0) {
perror("Reusing ADDR failed");
exit(1);
}
/*** END OF MODIFICATION TO ORIGINAL */
/* set up destination address */
memset(&addr,0,sizeof(addr));
addr.sin_family=AF_INET;
addr.sin_addr.s_addr=htonl(INADDR_ANY); /* N.B.: differs from sender
*/
addr.sin_port=htons(HELLO_PORT);
/* bind to receive address */
if (bind(fd,(struct sockaddr *) &addr,sizeof(addr)) < 0) {
perror("bind");
exit(1);
}
/* use setsockopt() to request that the kernel join a multicast
group */
mreq.imr_multiaddr.s_addr=inet_addr(HELLO_GROUP);
mreq.imr_interface.s_addr=htonl(INADDR_ANY);
if (setsockopt(fd,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq)) <
0) {
perror("setsockopt");
exit(1);
}
/* now just enter a read-print loop */
while (1) {
addrlen=sizeof(addr);
if ((nbytes=recvfrom(fd,msgbuf,MSGBUFSIZE,0,
   (struct sockaddr *) &addr,&addrlen)) < 0) {
perror("recvfrom");
exit(0);
}
puts(msgbuf);
}
}
```

Output:



5. Write a program to implement concurrent chat server that allows current logged in users to communicate with each other.

Server.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<sys/stat.h>


#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<unistd.h>

#include<netinet/in.h>
#include<arpa/inet.h>

void str_echo(int connfd,int port){
        int n,bufsize = 1024,len;
        char *buff = malloc(bufsize);
        struct sockaddr_in addr;
```

```
again: while((n=recv(connfd,buff,bufsize,0))>0){
                printf("From client connected to %d :",port);
                fputs(buff,stdout);
                printf("Reply to the client connected to %d :",port);
                fgets(buff,bufsize,stdin);
                send(connfd,buff,n,0);
        }


        if(n<0)
                goto again;
}
int main(){
        int listenfd,connfd,addrlen,pid;
        struct sockaddr_in address;
        if((listenfd = socket(AF_INET,SOCK_STREAM,0)) > 0)
                printf("The socket was created\n");
        else
                printf("Error in Socket creation\n");

        address.sin_family = AF_INET;
        address.sin_addr.s_addr = INADDR_ANY;
        address.sin_port= htons(15001);

        if( bind( listenfd,(struct sockaddr *)& address,sizeof(address)) ==0)
                printf("Binding Socket\n");
        else
                printf("ERROR in binding\n");

        if ((listen(listenfd, 3)) != 0){
            printf("Listen failed\n");
            exit(0);
        }
        else{
                getsockname(listenfd,(struct sockaddr *) &address,&addrlen);
                printf("Server listening on port  %d\n",address.sin_port);
        }


        for(;;){
                addrlen = sizeof(struct sockaddr_in);
                connfd = accept(listenfd,(struct sockaddr *)& address,&addrlen);
                if(connfd>0)
```

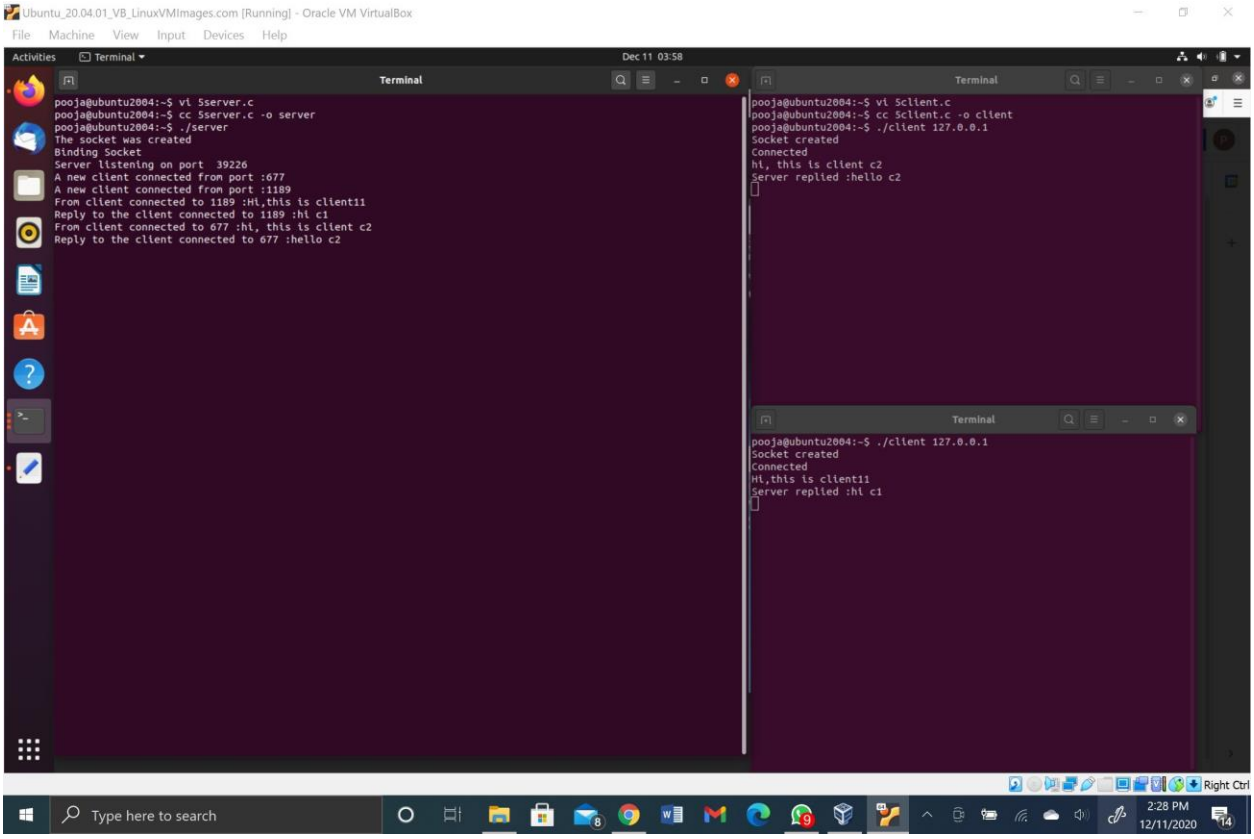
```
                        printf("A new client connected from port :%d \n",
address.sin_port);
                else
                        printf("A new client's connection wasn't accepted\n");

                if((pid=fork())==0){
                close(listenfd);
                str_echo(connfd,address.sin_port);
                exit(0);
                }

                close(connfd);
        }

        return 0;
}
```

Client.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<sys/stat.h>


#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>

#include<arpa/inet.h>
#include<netinet/in.h>

void str_cli(FILE *fp,int sockfd){
        int bufs=1024,cont;
        char *buff = malloc(bufs);

        while((fgets(buff,bufs,fp)!=NULL)){
                send(sockfd,buff,bufs,0);

                if((cont = recv(sockfd,buff,bufs,0)) >=0){
                        printf("Server replied :");
                        fputs(buff,stdout);
                }
        }
```

```
	printf("str cli\n");

	printf("\nEOF\n");
}

int main(int argc, char* argv[]){
	int create_socket,ret;

	struct sockaddr_in address;

	if((create_socket = socket(AF_INET,SOCK_STREAM,0)) >0)
		printf("Socket created\n");
	else
		printf("Socket creation error");

	address.sin_family = AF_INET;
	address.sin_port = htons(15001);

	inet_pton(AF_INET,argv[1],&address.sin_addr);

	if( (ret=connect(create_socket,(struct sockaddr *) &address,sizeof(address))) ==
0)
		printf("Connected\n");
	else
		printf("Error in connect");

	str_cli(stdin,create_socket);

	return 0;
}
```

Output:



6. Implementation of concurrent and iterative echo server using both connection and connectionless socket system call.

CON

Server.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <arpa/inet.h>

void str_echo(int connfd)
{
        int n;
        int bufsize = 300;
        char *buffer = malloc(bufsize);
again:
        while((n = recv(connfd, buffer, bufsize, 0)) > 0)
```

```
		send(connfd,buffer,n,0);

	if(n < 0)
		goto again;

	free(buffer);
}

int main()
{
	int listenfd, connfd, addrlen, pid, addrlen3;
	struct sockaddr_in address, cli_address;
	if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) > 0)
		printf("The socket was created\n");

	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons(15001);

	printf("The address before bind %s ...\n", inet_ntoa(address.sin_addr));

	if (bind(listenfd, (struct sockaddr *)&address, sizeof(address)) == 0)
		printf("Binding Socket\n");

	printf("The address after bind %s ...\n",inet_ntoa(address.sin_addr));

	listen(listenfd, 3);
	printf("Server is listening\n");

	getsockname(listenfd, (struct sockaddr *)&address, &addrlen3);
	printf("The server's local address %s ... and port %d\n",
inet_ntoa(address.sin_addr), htons(address.sin_port));

	for(;;)
	{
		addrlen = sizeof(struct sockaddr_in);
		connfd = accept(listenfd, (struct sockaddr *)&cli_address, &addrlen);
		int i = getpeername(connfd,(struct sockaddr *)&cli_address,&addrlen);

		if (connfd > 0)
			printf("The Client  %s is connected ... on port %d\n",
inet_ntoa(cli_address.sin_addr), htons(cli_address.sin_port));
```

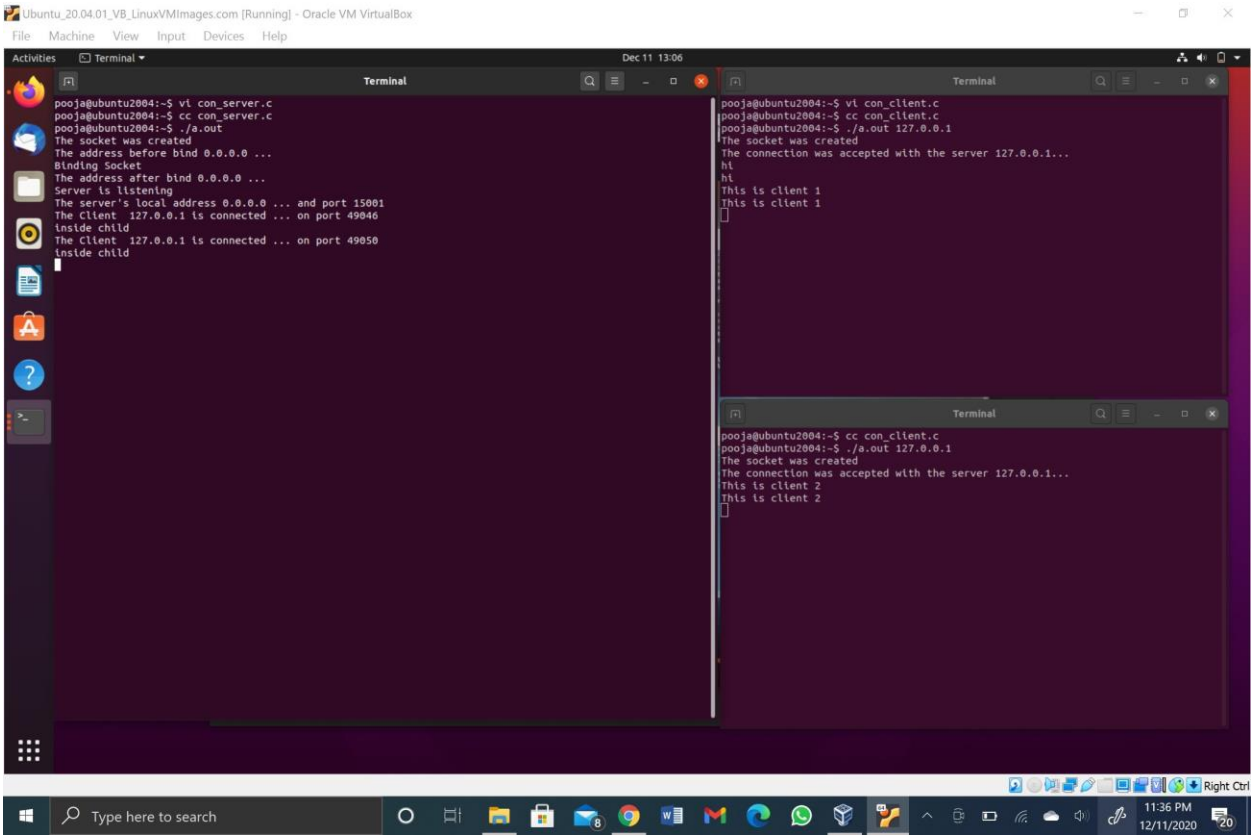
```
		if ((pid = fork()) == 0)
		{
			printf("inside child\n");
			close(listenfd);
			str_echo(connfd);
			exit(0);
		}

		close(connfd);
	}

	return 0 ;
}
```

Client.c

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

void str_cli(FILE *fp, int sockfd)
{
	int bufsize = 1024;
	char *buffer = malloc(bufsize);

	while (fgets(buffer, bufsize, fp) != NULL)
	{
		send(sockfd, buffer, sizeof(buffer), 0);
		if (recv(sockfd, buffer, bufsize, 0) > 0)
			fputs(buffer, stdout);
	}

	printf("\nEOF\n");

	free(buffer);
}


int main(int argc,char *argv[])
{
	int create_socket;
```

```
        struct sockaddr_in address;

        if ((create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0)
                printf("The socket was created\n");

        address.sin_family = AF_INET;
        address.sin_port = htons(15001);
        inet_pton(AF_INET, argv[1], &address.sin_addr);

        if (connect(create_socket, (struct sockaddr *)&address, sizeof(address)) == 0)
                printf("The connection was accepted with the server %s...\n",argv[1]);
        else
                printf("Error in connect\n");

        str_cli(stdin, create_socket);

        return close(create_socket);
}
```

Output:



ITR

Server.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
#include<arpa/inet.h>

void str_echo(int connfd)
{
        int n;
        int bufsize = 1024;
        char *buffer = malloc(bufsize);
        //printf("inside the function");
 again: while((n=recv(connfd, buffer, bufsize, 0))>0)
                send(connfd,buffer,n,0);
        if(n<0)
                goto again;
}

int main()
{
        int cont,listenfd,connfd,addrlen,addrlen2,fd,pid,addrlen3;

        //char fname[256];
        struct sockaddr_in address,cli_address;
        if ((listenfd = socket(AF_INET,SOCK_STREAM,0)) > 0)
                printf("The socket was created\n");
        address.sin_family = AF_INET;
        address.sin_addr.s_addr = INADDR_ANY;
        address.sin_port = htons(15001);
        printf("The address before bind %s  ...\n",inet_ntoa(address.sin_addr) );
        if (bind(listenfd,(struct sockaddr *)&address,sizeof(address)) == 0)
                printf("Binding Socket\n");
        printf("The address after bind %s  ...\n",inet_ntoa(address.sin_addr) );

        listen(listenfd,3);
        printf("server is listening\n");

        for(;;){
                addrlen = sizeof(struct sockaddr_in);
```

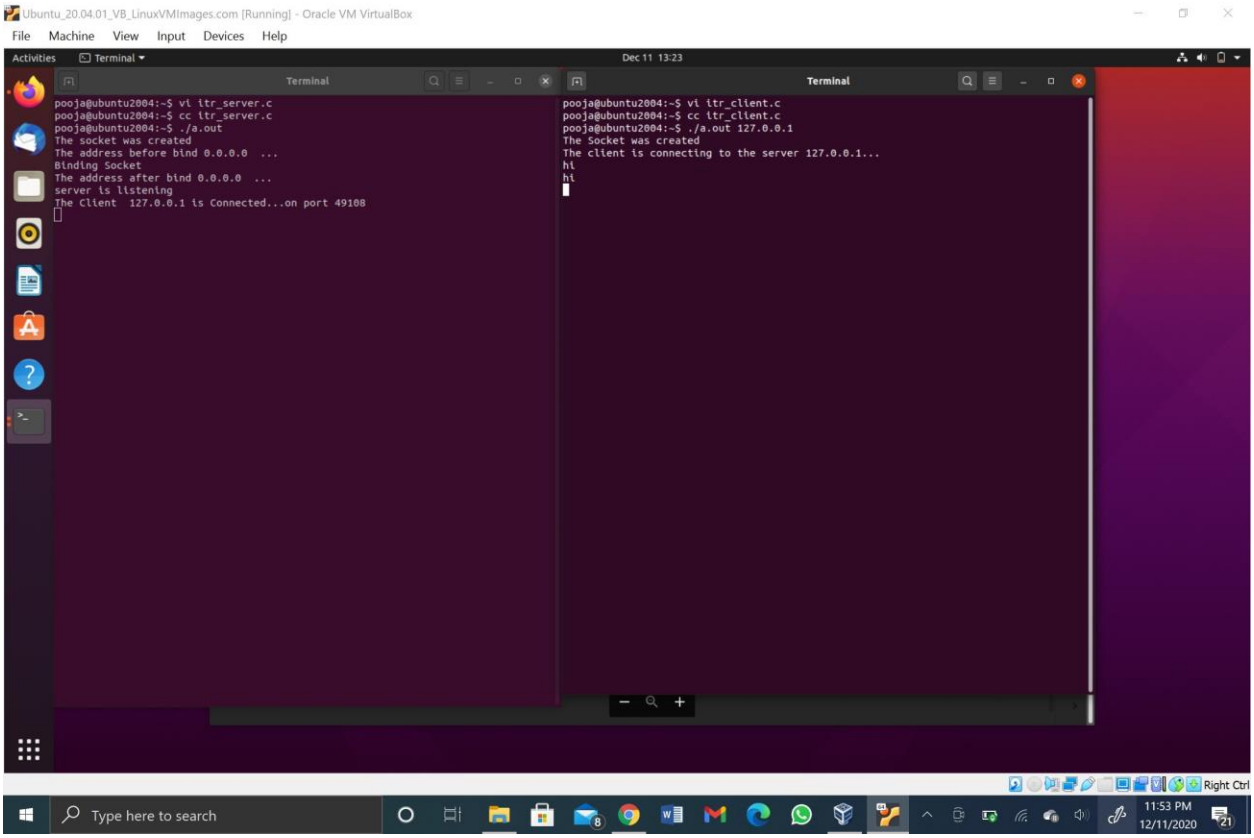
```
		connfd = accept(listenfd,(struct sockaddr *)&cli_address,&addrlen);
		/*if(connfd>0)
			printf("The client is connected\n");*/
		printf("The Client  %s is Connected...on port
%d\n",inet_ntoa(cli_address.sin_addr),htons(cli_address.sin_port));

		str_echo(connfd);
		close(connfd);
	}
	return 0 ;
}
```

Client.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
#include<arpa/inet.h>

void str_cli(FILE *fp, int sockfd)
{
	int bufsize = 1024, cont;
	char *buffer = malloc(bufsize);

	while(fgets(buffer,bufsize,fp)!=NULL)
	{
		send(sockfd, buffer, sizeof(buffer), 0);

		if((cont=recv(sockfd, buffer, bufsize, 0))>0) {
			fputs(buffer,stdout);
		}
	}
	printf("\nEOF\n");
}

int main(int argc,char *argv[])
{
	int create_socket;
	struct sockaddr_in address;
```

```
        if ((create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0)
        printf("The Socket was created\n");
        address.sin_family = AF_INET;
        address.sin_port = htons(15001);
        inet_pton(AF_INET,argv[1],&address.sin_addr);

        if (connect(create_socket,(struct sockaddr *) &address, sizeof(address)) == 0)
                printf("The client is connecting to the server %s...\n",argv[1]);
        else
                printf("error in connect \n");

        str_cli(stdin,create_socket);

        return close(create_socket);
}
```

Output:



ITR UDP
Server.c

```
#include<sys/types.h>
#include<sys/socket.h>
```

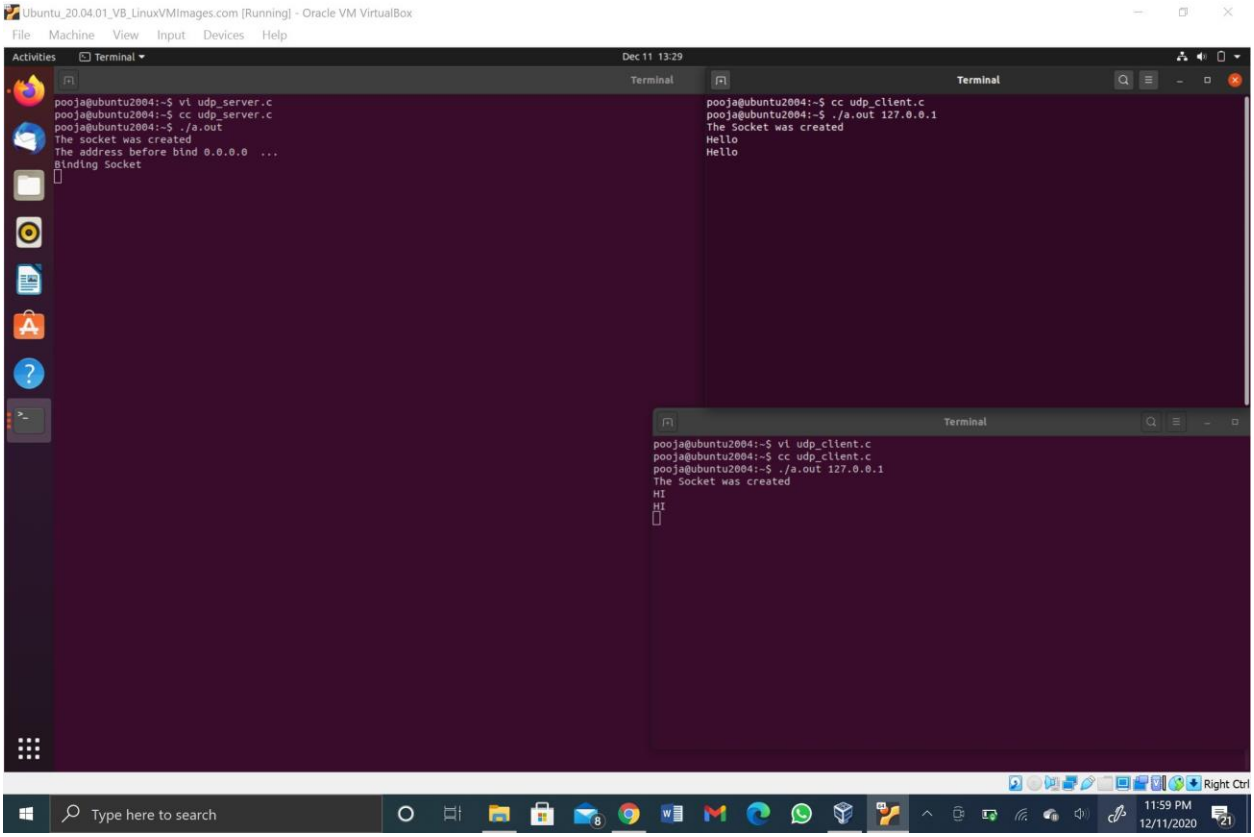
```
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
#include <arpa/inet.h>

void str_echo(int sockfd,struct sockaddr* cli_address, int clilen)
{
     int n;
     int bufsize = 1024;
     char *buffer = malloc(bufsize);
          int addrlen;

          for(;;){
          addrlen = clilen;
          n=recvfrom(sockfd,buffer,bufsize,0,cli_address,&addrlen); //recvfrom
          //printf("%s",buffer);
          sendto(sockfd,buffer,n,0,cli_address,addrlen);}                    //sendto
           //printf("%d n",n);


}
int main()
{
  int sockfd;
  struct sockaddr_in serv_address,cli_address;

  if ((sockfd = socket(AF_INET,SOCK_DGRAM,0)) > 0) //sockfd
    printf("The socket was created\n");

    serv_address.sin_family = AF_INET;
    serv_address.sin_addr.s_addr = INADDR_ANY;
    serv_address.sin_port = htons(16001);
    printf("The address before bind %s  ...\n",inet_ntoa(serv_address.sin_addr) );
    if (bind(sockfd,(struct sockaddr *)&serv_address,sizeof(serv_address)) == 0)     //bind
    printf("Binding Socket\n");
   str_echo(sockfd,(struct sockaddr *)&cli_address,sizeof(cli_address));

    return 0 ;
}
```

Client.c

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

void str_cli(FILE *fp, int sockfd,struct sockaddr* serv_address, int servlen)
{
   int bufsize = 1024, cont;
   char *buffer = malloc(bufsize);
            int addrlen = sizeof(struct sockaddr_in);
            while(fgets(buffer,bufsize,fp)!=NULL){

            sendto(sockfd, buffer, sizeof(buffer),0,serv_address,servlen);

            if((cont=recvfrom(sockfd, buffer, bufsize, 0,NULL,NULL)>0))
            {
            fputs(buffer,stdout);                //echo printing
            }}

  printf("\nEOF\n");
}
int main(int argc,char *argv[])
{
  int sockfd;

  //char fname[256];
  struct sockaddr_in serv_address;
  if ((sockfd = socket(AF_INET,SOCK_DGRAM,0)) > 0)
  printf("The Socket was created\n");
  serv_address.sin_family = AF_INET;
  serv_address.sin_port = htons(16001);

  inet_pton(AF_INET,argv[1],&serv_address.sin_addr);


  str_cli(stdin,sockfd,(struct sockaddr *)&serv_address,sizeof(serv_address));


  exit(0);
```

}

Output:



7. Implementation of remote control command execution using socket system call.

Server.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
#include<arpa/inet.h>
void remote_command(int connfd, int port)
{
        int n;
        int bufsize=1024;
        char *buffer=malloc(bufsize);
        do
        {
```

```
		while((n=recv(connfd, buffer, bufsize, 0))>0)
		{
			send(connfd, buffer, n, 0);
			printf("Port: %d\n",port);
			system(buffer);			//for reading commands
		}
	}while(n<0);		//EOF
}
int main()
{
	int cont, listenfd, connfd, addrlen, fd, pid;
	struct sockaddr_in address;
	if((listenfd=socket(AF_INET,SOCK_STREAM,0))>0)			//create socket
		printf("The socket was created.\n");
	address.sin_family=AF_INET;
	address.sin_addr.s_addr=INADDR_ANY;
	address.sin_port=htons(15001);
	if(bind(listenfd,(struct sockaddr*)&address,sizeof(address))==0)	//binding
socket
		printf("Binding Socket\n");
	listen(listenfd,3);				//listen
	printf("Server is listening.\n");
	for(;;)
	{
		addrlen=sizeof(struct sockaddr_in);
		connfd=accept(listenfd,(struct sockaddr*)&address,&addrlen);
	//accept
		if(connfd>0)
		{
			printf("The client %s is connected.\n",
			inet_ntoa(address.sin_addr));
		}
		if((pid=fork())==0)		//server forks in TCP
		{
			printf("Inside Child\n");
			close(listenfd);
			remote_command(connfd,htons(address.sin_port));
			exit(0);
		}
		close(connfd);
	}
	return 0;
}
```

Client.c

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
#include<arpa/inet.h>
void str_cli(FILE *fp, int sockfd)
{
	int bufsize=1024, cont;
	char *buffer=malloc(bufsize);
	while(fgets(buffer,bufsize,fp)!=NULL)
	{
		send(sockfd, buffer, sizeof(buffer),0);
		if((cont=recv(sockfd,buffer,bufsize,0))>0)
		{
			//fputs(buffer,stdout);
		}
	}
	printf("\nEOF\n");
}
int main(int argc, char *argv[])
{
	int create_socket;
	struct sockaddr_in address;
	if((create_socket=socket(AF_INET,SOCK_STREAM,0))>0)
		printf("The socket was created.\n");
	address.sin_family=AF_INET;
	address.sin_port=htons(15001);
	inet_pton(AF_INET, argv[1], &address.sin_addr);
	if(connect(create_socket,(struct sockaddr*)&address, sizeof(address))==0)
		printf("The connection was accepted with the server %s...\n",argv[1]);
	else
		printf("Error in connect()\n");
	str_cli(stdin, create_socket);
	return close(create_socket);
}
```

Output:

Ubuntu_20.04.01_VB_LinuxVMImages.com [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Dec 11 12:56

Terminal

```
pooja@ubuntu2004:~$ vi rmd_client.c
pooja@ubuntu2004:~$ cc -o cli rmd_client.c
pooja@ubuntu2004:~$ ./cli 127.0.0.1
The socket was created.
The connection was accepted with the server 127.0.0.1...
ls
ls-l
ls -l
ps
netstat
```

Terminal

```
pooja@ubuntu2004:~$ vi rmd_server.c
pooja@ubuntu2004:~$ cc -o serv try2_server.c
cc: error: try2_server.c: No such file or directory
cc: fatal error: no input files
compilation terminated.
pooja@ubuntu2004:~$ cc -o serv rmd_server.c
pooja@ubuntu2004:~$ ./serv
The socket was created.
Binding Socket
Server is listening.
The client 127.0.0.1 is connected.
Inside Child
Port: 49022
Sclient.c  a.out  client    clientpg2.c  Documents  dummy.txt  Pictures  rmd_client.c  rsa.cpp  serv
er    Templates
Sserver.c  cli    client.c  Desktop      Downloads  Music      Public    rmd_server.c  serv     serv
er.c  Videos
Port: 49022
sh: 1: ls-l: not found
Port: 49022
total 160
-rw-rw-r-- 1 pooja pooja  1022 Dec 11 03:55 Sclient.c
-rw-rw-r-- 1 pooja pooja  1762 Dec 11 03:54 Sserver.c
-rwxrwxr-x 1 pooja pooja 17240 Dec 11 03:15 a.out
-rwxrwxr-x 1 pooja pooja 17264 Dec 11 12:54 cli
-rwxrwxr-x 1 pooja pooja 17288 Dec 11 03:55 client
-rw-rw-r-- 1 pooja pooja  1221 Dec 11 03:15 client.c
-rw-rw-r-- 1 pooja pooja     0 Dec 10 09:19 clientpg2.c
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Desktop
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Documents
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Downloads
-rw-rw-r-- 1 pooja pooja     0 Dec 11 03:06 dummy.txt
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Music
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Pictures
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Public
-rw-rw-r-- 1 pooja pooja  1006 Dec 11 12:50 rmd_client.c
-rw-rw-r-- 1 pooja pooja  1370 Dec 11 12:51 rmd_server.c
-rw-rw-r-- 1 pooja pooja  2359 Dec 11 06:12 rsa.cpp
-rwxrwxr-x 1 pooja pooja 17344 Dec 11 12:53 serv
-rwxrwxr-x 1 pooja pooja 17504 Dec 11 03:54 server
-rw-rw-r-- 1 pooja pooja  1472 Dec 11 03:14 server.c
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Templates
drwxr-xr-x 2 pooja pooja  4096 Dec  8 06:31 Videos
Port: 49022
    PID TTY          TIME CMD
```

Right Ctrl

Type here to search 11:26 PM 12/11/2020

Ubuntu_20.04.01_VB_LinuxVMImages.com [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Dec 11 12:57

Terminal

```
pooja@ubuntu2004:~$ vi rmd_client.c
pooja@ubuntu2004:~$ cc -o cli rmd_client.c
pooja@ubuntu2004:~$ ./cli 127.0.0.1
The socket was created.
The connection was accepted with the server 127.0.0.1...
ls
ls-l
ls -l
ps
netstat
```

Terminal

```
   2525 pts/1    00:00:00 bash
   2890 pts/1    00:00:00 serv
   2900 pts/1    00:00:00 serv
   2911 pts/1    00:00:00 sh
   2912 pts/1    00:00:00 ps
Port: 49022
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 ubuntu2004:35922        maa05s06-in-f10.1:https TIME_WAIT
tcp        0      0 ubuntu2004:38478        maa03s36-in-f5.1e:https ESTABLISHED
tcp        0      0 ubuntu2004:55374        maa05s13-in-f14.1:https ESTABLISHED
tcp        0      0 localhost:15001         localhost:49022         ESTABLISHED
tcp        0      0 ubuntu2004:48512        maa05s09-in-f3.1e:https ESTABLISHED
tcp        0      0 ubuntu2004:55376        maa05s13-in-f14.1:https ESTABLISHED
tcp        0      0 localhost:49022         localhost:15001         ESTABLISHED
tcp        0      0 ubuntu2004:39054        172.217.194.189:https   ESTABLISHED
tcp        0      0 ubuntu2004:54364        74.125.24.189:https     ESTABLISHED
tcp        0      0 ubuntu2004:57828        ec2-52-32-204-93.:https ESTABLISHED
tcp        0      0 ubuntu2004:53614        maa03s31-in-f14.1:https ESTABLISHED
udp        0      0 ubuntu2004:bootpc       _gateway:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    30064    /run/user/1001/systemd/notify
unix  3      [ ]         DGRAM                    5766     /run/systemd/notify
unix  2      [ ]         DGRAM                    5782     /run/systemd/journal/syslog
unix  15     [ ]         DGRAM                    5792     /run/systemd/journal/dev-log
unix  8      [ ]         DGRAM                    5796     /run/systemd/journal/socket
unix  3      [ ]         STREAM     CONNECTED     33082
unix  3      [ ]         STREAM     CONNECTED     30218
unix  3      [ ]         STREAM     CONNECTED     21731
unix  3      [ ]         STREAM     CONNECTED     45201
unix  3      [ ]         STREAM     CONNECTED     33901    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     23531
unix  3      [ ]         STREAM     CONNECTED     23551
unix  3      [ ]         STREAM     CONNECTED     33986
unix  3      [ ]         STREAM     CONNECTED     33548    /run/user/1001/bus
unix  3      [ ]         STREAM     CONNECTED     20227    /run/systemd/journal/stdout
unix  2      [ ]         DGRAM                    33137
unix  3      [ ]         STREAM     CONNECTED     30241
unix  3      [ ]         STREAM     CONNECTED     45200
unix  3      [ ]         STREAM     CONNECTED     36428
unix  3      [ ]         STREAM     CONNECTED     35885    @/tmp/.ICE-unix/1643
unix  3      [ ]         STREAM     CONNECTED     33871    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     33588    /run/user/1001/bus
unix  3      [ ]         STREAM     CONNECTED     33190
```

Right Ctrl

Type here to search 11:27 PM 12/11/2020

8. Write a program to encrypt and decrypt the data using RSA and exchange the key securely using Diffeie-Hellman key exchange protocol.

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

#include<string.h>

long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;

char msg[100];

int prime(long int);

void ce();

long int cd(long int);

void encrypt();

void decrypt();

int main() {



        printf("\nENTER FIRST PRIME NUMBER\n");

        scanf("%d",&p);

        flag=prime(p);

        if(flag==0) {

                printf("\nWRONG INPUT\n");

                getch();

                exit(1);
```

```
}
printf("\nENTER ANOTHER PRIME NUMBER\n");
scanf("%d",&q);
flag=prime(q);
if(flag==0||p==q) {
        printf("\nWRONG INPUT\n");
        getch();
        exit(1);
}
printf("\nENTER MESSAGE\n");
fflush(stdin);
scanf("%s",msg);
for (i=0;msg[i]!=NULL;i++)
m[i]=msg[i];
n=p*q;
t=(p-1)*(q-1);
ce();
printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
for (i=0;i<j-1;i++)
printf("\n%ld\t%ld",e[i],d[i]);
encrypt();
```

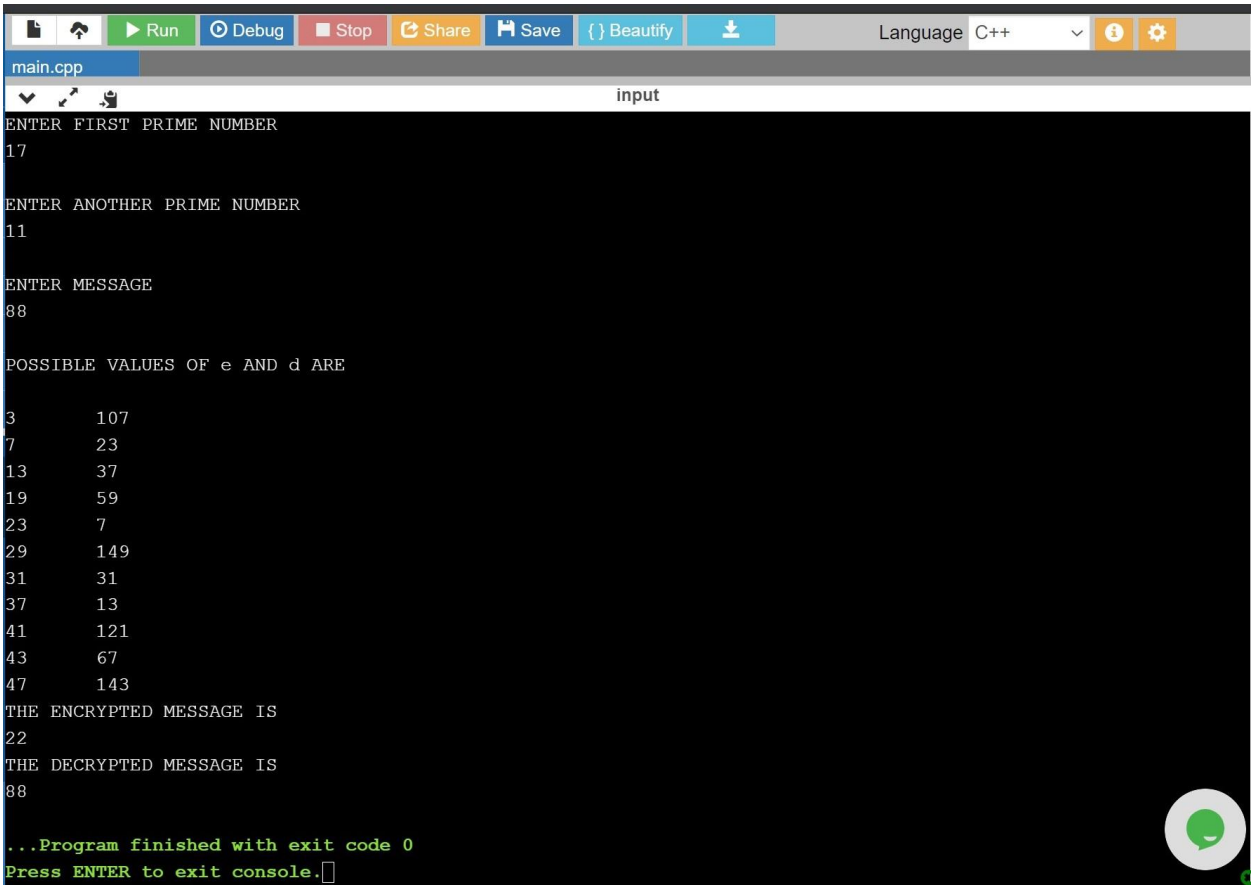
```
        decrypt();
        getch();
}
int prime(long int pr) {
        int i;
        j=sqrt(pr);
        for (i=2;i<=j;i++) {
                if(pr%i==0)
                    return 0;
        }
        return 1;
}
void ce() {
        int k;
        k=0;
        for (i=2;i<t;i++) {
                if(t%i==0)
                    continue;
                flag=prime(i);
                if(flag==1&&i!=p&&i!=q) {
                        e[k]=i;
                        flag=cd(e[k]);
```

```
                if(flag>0) {
                        d[k]=flag;
                        k++;
                }
                if(k==99)
                    break;
            }
        }
}
long int cd(long int x) {
        long int k=1;
        while(1) {
            k=k+t;
            if(k%x==0)
                return(k/x);
        }
}
void encrypt() {
        long int pt,ct,key=e[0],k,len;
        i=0;
        len=strlen(msg);
```

```
        while(i!=len) {
                pt=m[i];
                pt=pt-96;
                k=1;
                for (j=0;j<key;j++) {
                        k=k*pt;
                        k=k%n;
                }
                temp[i]=k;
                ct=k+96;
                en[i]=ct;
                i++;
        }
        en[i]=-1;
        printf("\nTHE ENCRYPTED MESSAGE IS\n");
        for (i=0;en[i]!=-1;i++)
        printf("%c",en[i]);
}
void decrypt() {
        long int pt,ct,key=d[0],k;
        i=0;
        while(en[i]!=-1) {
```

```
                ct=temp[i];
                k=1;
                for (j=0;j<key;j++) {
                        k=k*ct;
                        k=k%n;
                }
                pt=k+96;
                m[i]=pt;
                i++;
        }
        m[i]=-1;
        printf("\nTHE DECRYPTED MESSAGE IS\n");
        for (i=0;m[i]!=-1;i++)
        printf("%c",m[i]);
}
```

Output:



RSA:

```
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <string.h>
using namespace std;
long int gcd(long int a, long int b)
{
if(a == 0)
return b;
if(b == 0)
return a;
return gcd(b, a%b);
}
long int isprime(long int a)
{
int i;
for(i = 2; i < a; i++){
if((a % i) == 0)
return 0;
}
return 1;
}
```

```
long int encrypt(char ch, long int n, long int e)
{
int i;
long int temp = ch;
for(i = 1; i < e; i++)
temp = (temp * ch) % n;
return temp;
}
char decrypt(long int ch, long int n, long int d)
{

int i;
long int temp = ch;
for(i = 1; i < d; i++)
ch =(temp * ch) % n;
return ch;
}
int main()
{
long int i, len;
long int p, q, n, phi, e, d, cipher[50];
char text[50];
cout << "Enter the text to be encrypted: ";
cin.getline(text, sizeof(text));
len = strlen(text);
do {
p = rand() % 30;
} while (!isprime(p));
do {
q = rand() % 30;
} while (!isprime(q));
n = p * q;
phi = (p - 1) * (q - 1);
do {
e = rand() % phi;
} while (gcd(phi, e) != 1);
do {
d = rand() % phi;
} while (((d * e) % phi) != 1);
cout << "Two prime numbers (p and q) are: " << p << " and " << q <<
endl;
cout << "n(p * q) = " << p << " * " << q << " = " << p*q << endl;
cout << "(p - 1) * (q - 1) = "<< phi << endl;
cout << "Public key (n, e): (" << n << ", " << e << ")\n";
cout << "Private key (n, d): (" << n << ", " << d << ")\n";
for (i = 0; i < len; i++)
cipher[i] = encrypt(text[i], n, e);
cout << "Encrypted message: ";
for (i = 0; i < len; i++)
cout << cipher[i];
for (i = 0; i < len; i++)
```

```
text[i] = decrypt(cipher[i], n, d);
cout << endl;
cout << "Decrypted message: ";
for (i = 0; i < len; i++)
cout << text[i];
cout << endl;
return 0;
}
```

Output: