Name of Experiment: client-server communication Date:

Experiment No: 01 Experiment Result:

1. Implement client and Server Communication using Socket programming

file_server.c.

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.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))
        printf("socket created");
```

```
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");

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 connected to server\n",
                    inet_ntoa(address.sin_addr));
recv(new_socket,fname,255,0);
printf("request for filename %s received...\n",fname);

if((fd=open(fname,O_RDONLY))<0)
{
    char err[]="file doesnot exist";
    send(new_socket,err,size_of(err),0);
    perror("File Open failed");
}
```

```
while((cont=read(fd, buffer, bufsize))>0)
{
    send(new_socket, buffer, cont, 0);
}
printf("Request Completed \n");
close(new_socket);
return close(create_socket);
}
```

file_client.c

```
#include<stdio.h>
#include <stdlib.h>
#include< sys/types.h>
#include <sys/socket.h>
#include < unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int main( int argc, char* argv[])
{
int create_socket, cont, bufsize=1024;
char* buffer= malloc(bufsize);
char fname[256];
struct sockaddr_in address;

if((create_socket=socket(AF_INET, SOCK_STREAM,0))>0)
    printf("socket 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 accepted with server %s..\n",
                                        argv[1]);

printf("Enter the filename to Request:");
scanf("%s", fname);
send(create_socket, fname, sizeof(fname), 0);
printf("Request accepted... Recieving file...\n\n");

while((cont = recv(create_socket, buffer, bufsize, 0))>0)
{
        write(1, buffer, cont);
}
printf("\n EOF \n");
return close(create_socket);
}
```

Sample.txt

```
hi shreenidhi
how is nps going on?
```

output

```
gcc -o file_server file_server.c
./file_server
The socket was created
The client 127.0.0.1 connected
A request for filename
sample.txt received
Request completed
```

```
gcc -o file_client file_client.c
./client 127.0.0.1
The socket was created
The connection accepted
- with server 127.0.0.1
Enter the filename: sample.txt
hi shreenidhi
How is nps going on?
```

Name of Experiment: Distance vector Date: Page No. 5

Experiment No. 02 Experiment Result:

Q.2] write a program to implement distance vector routing algorithm for simple topology of routers.

```
#include <stdio.h>
int A[10][10], n, d[10], p[10];

void BellmanFord(int s) {
    int i, u, v;
    for(i=1; i<n; i++) {
        for(u=0; u<n; u++) {
            for(v=0; v<n; v++) {
                if (d[v] > d[u] + A[u][v]) {
                    d[v] = d[u] + A[u][v];
                    p[v] = u;
                }
            }
        }
    }

int main() {
    printf("Enter the number of vertices:");
    scanf("%d", &n);
    printf("Enter Adjacency matrix \n");
    int i, j;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d", &A[i][j]);
```

```
int source;
for(source=0; source<n; source++){
        for(i=0; i<n; i++){
                d[i]=999;
                p[i]=-1;
        }
        d[source]=0;
        Bellmanford(source);
printf("Router %d\n", source);
for(i=0; i<n; i++){
        if(i!=source){
                j=i;
                while(p[j]!=-1){
                        printf("%d <- ", j);
                        j=p[j];
                }
        printf("%d \t cost %d \n\n", source, d[i]);
}
return 0;
}
```

Name of Experiment: Hamming.c Date: Page No. 07

Experiment No. 03 (a) Experiment Result:

Q. write a progeam to implement error detection & correction concept using checksum & hamming code

```
#include<stdio.h>
#include <stdlib.h>
void main() {
    int data[7], rec[7], i, r1, r2, r3, c;
    printf("Enter the message bit one by one (4-bit): ");
    scanf("%d %d %d %d", &data[0], &data[1], &data[2],
                                        &data[4]);

    data[6] = data[0] ^ data[2] ^ data[4];
    data[5] = data[0] ^ data[1] ^ data[4];
    data[3] = data[0] ^ data[1] ^ data[2];
    printf("encoded bit's are given below:\n");
    for(i=0; i<7; i++) {
        scanf("%d", &rec[i]);
    }

    r1 = rec[6] ^ rec[4] ^ rec[2] ^ rec[0];
    r2 = rec[5] ^ rec[4] ^ rec[1] ^ rec[0];
    r3 = rec[3] ^ rec[2] ^ rec[1] ^ rec[0];
    c = r3 *4 + r2 * 2 + r1;

    if(c==0) {
        printf("congratualations there is
                                no error\n");
    } else {
        printf("\n error in the position : %d \n
the current message is \n", c);
```

```
if (rec[7-c] == 0) {
        rec[7-c] = 1;
} else {
        rec[7-c] = 0;
}
for (i=0; i<7; i++) {
        printf("%d", rec[i]);
    }
}
```

Name of Experiment: checksum.c Date: Page No. 09

Experiment No. 03 (b) Experiment Result:

```
#include<stdio.h>
unsigned fields[10];

unsigned short checksum(){
    int i;sum=0;
    printf("Ip header info in 16 bits : ");
    for(i=0;i<9;i++){
        printf("Field %d\n", i+1);
        scanf("%x", &field[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;
    result1 = checksum();
    printf("computed checksum at sender %x\n",result1);

    result2 = checksum();
    printf("computed checksum at reciever %x",
    result2);
```

```
if(result1 == result2) {
        printf("No Error\n");
}
else {
        printf("Error Detected\n");
}
}
```

Output

```
gcc checksum.c
./a.out

Enter the IP header info in 16 bit words.
Field 1
4501
Field 2
003C
Field 3
1C46
Field 4
4000
Field 5
4006
Field 6
ac10
Field 7
0a63
Field 8
ac10
Field 9
0a0c.

computed checksum at reciever bles
```

Enter IP header info in 16 bit words

Field1
4501
Field2
003C
Field3
1C46
Field4
4000
Field5
4006
Field6
ac10
Field7
0a63
Field8
ac10
Field9
0a0c

Computed checksum at receiver bies

No Error.

Name of Experiment: multicast routing Date: Page No. 11

Experiment No: 04 Experiment Result:

Q.4] Implement a simple multicast routing mechanism.

```
//Listener
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "255.0.0.37"
#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;

  if((fd=socket(AF_INET, SOCK_DGRAM,0))<0){
        perror("socket creation failed\n");
        exit(1);}

  if(setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &yes,
                sizeof(yes))<0){
    perror("Reusing addr failed"); exit(1); }
```

```
memset (&addr, 0, sizeof (addr));
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = htonl (INADDR_ANY);
addr.sin_port = htons( HELLO_PORT);

if(bind(fd, (struct sockaddr *)&addr, sizeof(addr))<0)
{ perror("bind");
  exit(1);
}

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("set sockopt\n"); exit(1);
    }
while(1) {
    addrlen = sizeof(addr);
    if ((nbytes = recvfrom(fd, msgbuf, MSGBUFSIZE,0,
          (struct sockaddr *)&addr, &addrlen))<0){
        perror("recv from\n");
      }
      puts(msgbuf);
    }
}
```

Name of Experiment..... Date..... Page No. 13

Experiment No..... Experiment Result.....

```
//sender
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "255.0.0.37"

int main(int argc, char* argv[])
{
    struct sockaddr_in addr;
    int fd, cnt;
    struct ip_mreq mreq;
    char *message = "RVCE-CSE";
    if((fd=socket(AF_INET, SOCK_DGRAM, 0))<0){
        perror("socket creation failed\n");
        exit(1); }
    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);
```

```
while(1){
  if(sendto(fd, message, sizeof(message), 0,
        (struct sockaddr *)&addr, sizeof(addr)<0)
  {
    perror("send to error\n");
    exit(1);
  }
  sleep(1);
}
```

output.

```
gcc -o sender sender.c
./sender

gcc -o listener listener.c
./listener
RVCE - CSE
RVCE - CSE
RVCE - CSE
RVCE - CSE

./ listener
RVCE - CSE
RVCE - CSE
RVCE - CSE
RVCE - CSE
```

Name of Experiment: chat server Date: Page No. 15

Experiment No: 05 Experiment Result:

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

Server.c

```
#include<stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#define max 80

void str_send (int sockfd1, int sockfd2)
{
    char buff[MAX];
    for(;;){
        bzero(buff, MAX);
        read (sockfd1, buff, sizeof (buff));
        printf(" client1 : %s\n", buff);
        write(sockfd2, buff, sizeof (buff));
        bzero(buff, MAX);

        read (sockfd2, buff, sizeof (buff));
        printf (" from client2: %s\n", buff);
        write( sockfd1, buff, sizeof (buff));
        bzero(buff, MAX);
        if (strcmp(" exit", buff, 4) ==0){
            printf(" server exit...\n");
            break; } } }
```

```
int main(int argc, char* argv[])
{ int cont, listenfd, connfd1, connfd2, addrlen1, addrlen2,
        fd, pid, addrlen3, i1, i2;
  struct sockaddr_in address, cli_adress1, cli_adress2;
  if((listenfd=socket(AF_INET, SOCK_STREAM, 0))>0)
        printf("The socket was created\n");
  address.sin_family = AF_INET;
  inet_pton(AF_INET, argv[2], &address.sin_addr);
  address.sin_port = htons(atoi(argv[1]));

  if(bind(listenfd, (struct sockaddr *) &address,
                sizeof(address))==0)
        printf("The adress after bind is %s.\n",
                inet_ntoa(adress.sin_addr));
  listen(listenfd, 3);

  printf("server is listening\n");

  for(;;){
        addrlen1 = sizeof(struct sockaddr_in);
        addrlen2 = sizeof(struct sockaddr_in);
        connfd1 = accept(listenfd, (struct sockaddr *) &
                cli_adress1, &addrlen1);
        connfd2 = accept(listenfd, (struct sockaddr *) &cli_adress2,
                &addrlen2);
        if((pid=fork())==0){ close(listenfd);
                str_send(connfd1, connfd2); exit(0); }
        close(connfd1); close(connfd2); return 0; }
```

Name of Experiment..... Date..... Page No. 17

Experiment No..... Experiment Result.....

client.c

```
#include <stdio.h>
#define MAX 10
int main(int argc, char *argv[]) {
  int create_socket, n;
  char buff[MAX];
  struct sockaddr_in address;

  if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
        printf("socket created");
  address.sin_family = AF_INET;
  address.sin_port = htons(atoi(argv[1]));
  inet_pton(AF_INET, argv[2], &address.sin_addr);

  if(connect(create_socket, (struct sockaddr*)&address,
                sizeof(address))==0)
      printf("connection was accepted with server %s",
                                argv[1]);
  else
    for(;;) {
        bzero(buff, sizeof(buff));
        printf("Enter string:");
        n=0; while((buf[n++] = getchar()) != '\n');
        write(create_socket, buf, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(create_socket, buff, sizeof(buff));
        printf("From other client: %s", buff);
    if(strncmp(buff, "exit", 4))==0) { printf("client exit");
      break; } }
  return close(create_socket);
```

client2.c

```
#include<stdio.h>
#define MAX 10
int main(int argc, char *argv[]){
int n,create_socket; char buff[MAX];
struct sockaddr_in adress;
if ((create_socket = socket(AF_INET, SOCK_STREAM, 0))>0)
        printf("socket created");
address.sin_family = AF_INET;
adress.sin_port = htons(atoi(argv[1]));
inet_pton(AF_INET, argv[2], &address.sin_addr);
if (connect(create_socket, (struct sockaddr *)&adress,
        sizeof(address)) ==0){printf("connection accepted"); }
else
    for(;;){
        bzero(buff, sizeof(buff));
        read(create_socket, buff, sizeof(buff));
        printf("From other client: %s ",buff);
        if((strncmp(buff, "exit", 4)) ==0){
            printf("client exit \n"); break; }
        bzero(buff, sizeof(buff));
        printf("Enter the string :");
        n=0; while((buff[n++]=getchar()) != '\n');
        write(create_socket, buff, sizeof(buff));
        }
    return close(create_socket);
}
```

output:

```
gcc -o client1 client.c
gcc -o client2 client.c
gcc -o server server.c

./server 5000 127.0.0.1
 socket created
client1 : hi
client2 : hello

./client1 5000 127.0.0.1
Socket created
connection accepted with Server at poet 5000
------ CHAT SERVER -----
 Enter the string : hi
 From other client : hello.

./client2 5000 127.0.0.1
socket created
connection accepted with server at poet 5000
------ CHAT SERVER -----
 From other client : hi
 Enter the string : hello
```

Name of Experiment..... Date..... Page No. 19

Experiment No. 06 Experiment Result.....

6] Implementation of concurrent and iterative echo-server that allows concurrent by using both connection and connectionless socket system calls.

UDP - iterative server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <time.h>
#define MAXLINE 5555
int main(int argc, char * argv[]){
  time_t clk = time(NULL);
  int sockfd, n; unsigned short port;
  socklen_t len; char mesg[1024];
  struct sockaddr_in servaddr, cliaddr;
  port = (unsigned short) atoi(argv[1]);
  if((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
      printf("error in socket creation\n");
  bzero(&servaddr, sizeof(servaddr));
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servaddr.sin_port = htons(port);
  len = sizeof(servaddr);
  if(bind(sockfd, (struct sockaddr*)&servaddr, len) < 0)
      perror("bind\n");
```

```
for(;;) {
    len = sizeof(cliaddr);
    n = recvfrom(sockfd, &mesg, MAXLINE, 0,
        (struct sockaddr *)&cliaddr, &len);
    struct sockaddr_in client_addr = cliaddr;
    sendto(sockfd, &mesg, sizeof(mesg), 0,
        (struct sockaddr *)&cliaddr, len);
    printf("message sent to client successfully\n");
}
```

client.c UDP.

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#define MAXLINE 5566.
int main(int argc, char * argv[])
{
    int sockfd, n, len; char mesg[1024];
    unsigned short port;
    char sendline[MAXLINE], recvline[MAXLINE];
    struct sockaddr_in servaddr;
    if((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
        perror("socket");
    printf("socket created successfully\n");
    bzero(&servaddr, sizeof(servaddr));
```

```
servaddr.sin_family = AF_INET;
port = (unsigned short) atoi(argv[1]);
servaddr.sin_port = htons(port);
if (inet_pton (AF_INET, argv[2], &servaddr.sin_addr) < 0)
{
    perror("inet_pton");
}
len = sizeof(servaddr);
printf("Enter the msg to be sent: ");
fgets(sendline, MAXLINE, stdin);
sendto(sockfd, &sendline, strlen(sendline), 0,
        (struct sockaddr *) &servaddr, len);
printf("message sent to server\n");
n = recvfrom(sockfd, &recvline, MAXLINE, 0,
        (struct sockaddr *) &servaddr, &len);
recvline[n] = '\0';
printf("SERVER: ");
printf("%s", recvline);
printf("\n-------- -----\n");
}
```

output

```
gcc -o server server.c
gcc -o client client.c

./server 4455
server started ----
binding successfull ----
message sent to the client successfully...

./client 4455 127.0.0.1
socket was created succesfully
Enter the message to be sent: hi
message sent server -----
SERVER : hi
```

echo-server.c concurrent TCP

```
#include<sys/types.h>
#include<stdio.h>
#include<sys/socket.h>
#include<fcntl.h>
#include<arpa/inet.h>
void str_echo(int connfd)
{ int n, buffsize=1024;
  char* buffer = malloc(bufsize);

again: while((n=recv(connfd, buffer, bufsize, 0))>0)
          send(connfd, buffer, n, 0);
       if(n<0)
          goto again;
}
int main(){
   int const, listenfd, connfd, addrlen, fd, pid;
   struct sockaddr_in sockaddress;
  if((listenfd=socket(AF_INET, SOCK_STREAM, 0))>0)
          printf("socket 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)
          printf("binding socket\n");
   listen(listenfd, 3);
```

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

    if (connfd>0)
    {printf("client %s is connected\n",
                inet_ntoa(address.sin_addr));
    }
    if ((pid = forkc())==0){
            printf("inside child\n");
            close (listenfd);
            str_echo (connfd);
            exit(0);
        }
close(connfd);
return 0; }
```

echo-server.c (iterative) (TCP)

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

```
void str-echo (int connfd)
{ int n, bufsize=1024;
  char* buffer= malloc(bufsize);
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, fd, pid;
struct sockaddr_in address;
if((listenfd=socket(AF-INET, SOCK-STREAM,0))>0)
          printf("socket created\n");
address.sin-family = AF-INET;
address.sin-addr.s-addr = INADDR-ANY;
address.sin-port = htons(15004);
if (bind(listenfd, (struct sockaddr*) &address,
                  sizeof(address))==0)
       printf("binding Socket\n");
listen(listenfd, 3);
for(;;) {
addrlen = sizeof(struct sockaddr_in);
connfd = accept(listenfd, (struct sockaddr*) &address,
                  sizeof(address)) &addrlen);
if(connfd>0){
       printf("The client %.s connected", inet-ntoa(address.
                                          sin-addr));
}
```

```
printf("inside child\n");
str_echo(connfd);
close(connfd);
return 0;
}
```

echo_client.c — client code is same for both iterative & concurrent server.

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdlib.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("Enter message to be sent and echoed\n");
  }
  printf("\n EOF \n");
}
```

```
int main(int argc, char * argv[]){
    int create-socket;
    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(15001);
  inet-pton(AF-INET, argv[1], &address.sin-addr);

  if(connect(create-socket, (struct sockaddr *)&address,
            sizeof(address))==0){
  printf("Enter msg to be Sent & echoed\n");
   str-cli(stdin, create-socket);
   return close(create-socket);
}
```

Output (concurrent)

```
gcc -o server echo-server.c
./server
socket created
binding successfull
server listening
The client 127.0.0.1 connected
The client 127.0.0.1 connected
```

client-1

```
./client 127.0.0.1
socket created.
Enter message: Hi
Hi
```

client-2

```
./client 127.0.0.1
socket created
Enter message: Hi
Hi
```

output (iterative)

```
gcc -o echo-server echo-server.c
./echo-server
The socket was created
Binding socket
server is listening
The client 127.0.0.1 connected
```

```
gcc -o client client.c
./client 127.0.0.1
The socket was created
Enter the message to be sent & echoed:
hi
hi
```

Name of Experiment: Remote command Date: Page No. 27

Experiment No. 07 Experiment Result:

4] implement remote command execution using socket system call.

Server.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<errno.h>
int main(){
 int sd, acpt, len, bytes, port;
 char send[50], receiv[50];
 struct sockaddr_in serv, cli;
 if((sd=socket(AF_INET, SOCK_STREAM, 0))<0)
 {  printf("error in socket\n");
    exit(0);
 }
 bzero(&serv, sizeof(serv));
 serv.sin_family = AF_INET;
 serv.sin_port = htons(15002);
 serv.sin_addr.s_addr=htonl(INADDR_ANY);
if(bind(sd, (struct sockaddr*)&serv,
                    sizeof(serv))<0)
{ printf("error in bind\n"); exit(0); }
listen(fd, 3);
act = accept(sd, (struct sockaddr*) NULL, NULL);
```

```
while(1){
    bytes = recv(acpt, receiv, 50, 0);
    receiv[bytes] = '\0';
    if(strcmp(receiv, "end") == 0)
    {
        close(acpt);
        close(sd);
        exit(0);
    }
    else{
        printf("command received %s", receiv);
        system(receiv);
        printf("\n");
    }
}
}
```

client.c

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

int main(){
    int sd, acpt, len, bytes, port;
    char send[50], receiv[50];
    struct sockaddr_in serv, cli;
```

```
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) < 0)
{
    printf ("Error in socket\n");
    exit(0);
}
bzero (&serv, sizeof (serv));

serv.sin_family = AF_INET;
serv.sin_port = htons (15002);
serv.sin_addr.s_addr = htonl(INADDR_ANY);
if (connect(sd, (struct sockaddr *)&serv, sizeof (serv)) < 0)
{ printf(" error in Connection\n");
    exit(0);
}
while(1){
    printf("Enter the command : ");
    gets(send1);
    if (strcmp(send1, "end") != 0)
    {   send(sd, send1, 50, 0);
    }
    else { send (sd, send1, 50, 0);
        close(sd);
        break;
    }
}
}
```

output:

```
gcc -o server server.c
./server
command received : ls
client client.c server server.c

gcc -o client client.c
./client 127.0.0.1
Enter the command: ls
```

Name of Experiment: RSA. Date:

Experiment No: 08 Experiment Result:

8) write a program to encrypt and decrypt the data using RSA and exchange the key securely using Diffie-hellman key exchange protocol.

rsa.cpp

```
#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, 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 no. p&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;
}
```

Diffie hellman

Dh.c

```
#include<stdio.h>
int compute (int a, int m, int n){
    int r, y=1;
    while (m>0){
        r=m%2;
        if(r==1)
            y=(y*a)%n;
        a=a*a%n;
        m=m/2;
    }
    return y;
}

int main(){
    int p=23;
    int g=5;
    int a,b;
    int A,B;
```

```
a = rand();
A = compute (g, a, p);

b = rand();
B = compute (g, b, p);

int keyA = compute (B, a, p);
int keyB = compute (A, b, p);

printf("\n Alice secret key is %d \n Bob's Secret
        key is %d \n", keyA, keyB);

return 0;
}
```