Practical No 1. Write an echo program with client and iterative server using TCP.

```
//TCP Echo Client
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>

#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/

int
main(int argc, char **argv)
{
 int sockfd;
 struct sockaddr_in servaddr;
 char sendline[MAXLINE], recvline[MAXLINE];

 //basic check of the arguments
 //additional checks can be inserted
 if (argc !=2) {
  perror("Usage: TCPClient <IP address of the server");
  exit(1);
 }

 //Create a socket for the client
 //If sockfd<0 there was an error in the creation of the socket
 if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
 perror("Problem in creating the socket");
  exit(2);
 }

 //Creation of the socket
 memset(&servaddr, 0, sizeof(servaddr));
 servaddr.sin_family = AF_INET;
 servaddr.sin_addr.s_addr=  inet_addr(argv[1]);
 servaddr.sin_port =  htons(SERV_PORT); //convert to big-endian order

 //Connection of the client to the socket
 if (connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr))<0) {
  perror("Problem in connecting to the server");
  exit(3);
```

```
 }

 while (fgets(sendline, MAXLINE, stdin) != NULL) {

  send(sockfd, sendline, strlen(sendline), 0);

  if (recv(sockfd, recvline, MAXLINE,0) == 0){
   //error: server terminated prematurely
   perror("The server terminated prematurely");
   exit(4);
  }
  printf("%s", "String received from the server: ");
  fputs(recvline, stdout);
 }

 exit(0);
}
```

//TCP Iterative Server

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

#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/
#define LISTENQ 8 /*maximum number of client connections */

int main (int argc, char **argv)
{
 int listenfd, connfd, n;
 socklen_t clilen;
 char buf[MAXLINE];
 struct sockaddr_in cliaddr, servaddr;

 //creation of the socket
 listenfd = socket (AF_INET, SOCK_STREAM, 0);

 //preparation of the socket address
 servaddr.sin_family = AF_INET;
 servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
 servaddr.sin_port = htons(SERV_PORT);
```

```
bind (listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

listen (listenfd, LISTENQ);

printf("%s\n","Server running...waiting for connections.");

for ( ; ; ) {

 clilen = sizeof(cliaddr);
 connfd = accept (listenfd, (struct sockaddr *) &cliaddr, &clilen);
 printf("%s\n","Received request...");

 while ( (n = recv(connfd, buf, MAXLINE,0)) > 0)  {
  printf("%s","String received from and resent to the client:");
  puts(buf);
  send(connfd, buf, n, 0);
 }

if (n < 0) {
 perror("Read error");
 exit(1);
}
close(connfd);

}
//close listening socket
close (listenfd);
}
```

Prac No: 02 Write an echo program with client and concurrent server using UDP.

// Client program – ClientEcho.java

```
import java.net.*;
import java.util.*;

public class ClientEcho
{

  public static void main( String args[] ) throws Exception
  {
    InetAddress add = InetAddress.getByName("snrao");
    DatagramSocket dsock = new DatagramSocket( );
    String message1 = "This is client calling";
    byte arr[] = message1.getBytes( );
    DatagramPacket dpack = new DatagramPacket(arr, arr.length, add, 7);
    dsock.send(dpack);                          // send the packet
    Date sendTime = new Date();                 // note the time of sending the message
    dsock.receive(dpack);                       // receive the packet
    String message2 = new String(dpack.getData( ));
    Date receiveTime = new Date( );  // note the time of receiving the message
    System.out.println((receiveTime.getTime( ) - sendTime.getTime( )) + " milliseconds echo time for "
+ message2);
  }
}
```

//Client program – ServerEcho.java

```
import java.net.*;
import java.util.*;
public class ServerEcho
{
  public static void main( String args[]) throws Exception
  {

    DatagramSocket dsock = new DatagramSocket(7);
    byte arr1[] = new byte[150];
    DatagramPacket dpack = new DatagramPacket(arr1, arr1.length );
    while(true)
    {
      dsock.receive(dpack);
      byte arr2[] = dpack.getData();
      int packSize = dpack.getLength();
      String s2 = new String(arr2, 0, packSize);
      System.out.println( new Date( ) + "  " + dpack.getAddress( ) + " : " + dpack.getPort( ) + " "+ s2);
      dsock.send(dpack);

    } }         }
```

Practical No: 3. Write an echo program with client and concurrent server using TCP.

// TCP Echo Client

```
#include            <stdlib.h>
#include             <stdio.h>
#include       <sys/types.h>
#include      <sys/socket.h>
#include       <netinet/in.h>
#include            <string.h>
#include <arpa/inet.h>
#define MAXLINE 4096 /*max text line length*/ #define
SERV_PORT 3000 /*port*/
Int main(int argc, char **argv)
 {
  int sockfd;
  struct sockaddr_in servaddr;
  char sendline[MAXLINE], recvline[MAXLINE];
  //basic check of the arguments
  //additional checks can be inserted if
  (argc !=2) {
   perror("Usage: TCPClient <IP address of the server");
   exit(1);
  }
//Create a socket for the client
  //If sockfd<0 there was an error in the creation of the socket if
  ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
  perror("Problem in creating the socket");
   exit(2);
  }
//Creation of the socket memset(&servaddr, 0,
sizeof(servaddr));       servaddr.sin_family       =
AF_INET;
  servaddr.sin_addr.s_addr= inet_addr(argv[1]);
  servaddr.sin_port =  htons(SERV_PORT); //convert to big-endian order
//Connection of the client to the socket
  if (connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr))<0) {
   perror("Problem in connecting to the server");
   exit(3);
  }
while (fgets(sendline, MAXLINE, stdin) != NULL) { send(sockfd,
sendline, strlen(sendline), 0);
if (recv(sockfd, recvline, MAXLINE,0) == 0){
    //error: server terminated prematurely perror("The
    server terminated prematurely"); exit(4);
   }
printf("%s", "String received from the server: ");
fputs(recvline, stdout);
  }


  exit(0);}
```

//TCP Concurrent Echo Server

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

#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/
#define LISTENQ 8 /*maximum number of client connections*/

int main (int argc, char **argv)
{
 int listenfd, connfd, n;
 pid_t childpid;
 socklen_t clilen;
 char buf[MAXLINE];
 struct sockaddr_in cliaddr, servaddr;

 //Create a socket for the soclet
 //If sockfd<0 there was an error in the creation of the socket
 if ((listenfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
 perror("Problem in creating the socket");
  exit(2);
 }


 //preparation of the socket address
 servaddr.sin_family = AF_INET;
 servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
 servaddr.sin_port = htons(SERV_PORT);

 //bind the socket
 bind (listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

 //listen to the socket by creating a connection queue, then wait for clients
 listen (listenfd, LISTENQ);
```

```
printf("%s\n","Server running...waiting for connections.");

for ( ; ; ) {

 clilen = sizeof(cliaddr);
 //accept a connection
 connfd = accept (listenfd, (struct sockaddr *) &cliaddr, &clilen);

 printf("%s\n","Received request...");

 if ( (childpid = fork ()) == 0 ) {//if it's 0, it's child process

 printf ("%s\n","Child created for dealing with client requests");

 //close listening socket
 close (listenfd);

 while ( (n = recv(connfd, buf, MAXLINE,0)) > 0)  {
  printf("%s","String received from and resent to the client:");
  puts(buf);
  send(connfd, buf, n, 0);
 }

 if (n < 0)
  printf("%s\n", "Read error");
 exit(0);
}
//close socket of the server
close(connfd);
}
}
```

Practical No: 4. Write an echo program with client and concurrent server using UDP.

//Client program – ClientEcho.java

```
import java.net.*;
import java.util.*;


public class ClientEcho
{
  public static void main( String args[] ) throws Exception
 {


    InetAddress add = InetAddress.getByName("snrao");


    DatagramSocket dsock = new DatagramSocket( );

    String message1 = "This is client calling";

    byte arr[] = message1.getBytes( );

    DatagramPacket dpack = new DatagramPacket(arr, arr.length, add, 7);

    dsock.send(dpack);                          // send the packet

    Date sendTime = new Date();                    // note the time of sending the message

    dsock.receive(dpack);                        // receive the packet

    String message2 = new String(dpack.getData( ));

    Date receiveTime = new Date( );  // note the time of receiving the message

    System.out.println((receiveTime.getTime( ) - sendTime.getTime( )) + " milliseconds echo time for "
+ message2);


  }


}
```

// **Client program – ServerEcho.java**

```
import java.net.*;
import java.util.*;

public class ServerEcho

{


  public static void main( String args[]) throws Exception
  {

    DatagramSocket dsock = new DatagramSocket(7);
    byte arr1[] = new byte[150];
    DatagramPacket dpack = new DatagramPacket(arr1, arr1.length );

    while(true)
    {

      dsock.receive(dpack);

      byte arr2[] = dpack.getData();

      int packSize = dpack.getLength();

      String s2 = new String(arr2, 0, packSize);

      System.out.println( new Date( ) + "  " + dpack.getAddress( ) + " : " + dpack.getPort( ) + " "+ s2);
      dsock.send(dpack);

    }
  }
}
```

Practical No : 5. Write a client and server program for chatting.

//**Client program: GossipClient.java**

```
import java.io.*;
import java.net.*;

public class GossipClient
{

  public static void main(String[] args) throws Exception
 {


    Socket sock = new Socket("127.0.0.1", 3000);

                        // reading from keyboard (keyRead object)

    BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

                        // sending to client (pwrite object)

    OutputStream ostream = sock.getOutputStream();
    PrintWriter pwrite = new PrintWriter(ostream, true);


                        // receiving from server ( receiveRead  object)

    InputStream istream = sock.getInputStream();

    BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

    System.out.println("Start the chitchat, type and press Enter key");

    String receiveMessage, sendMessage;


    while(true)
    {

      sendMessage = keyRead.readLine();  // keyboard reading
      pwrite.println(sendMessage);       // sending to server
      pwrite.flush();                    // flush the data

      if((receiveMessage = receiveRead.readLine()) != null) //receive from server
      {
         System.out.println(receiveMessage); // displaying at DOS prompt
      }
    }    } }
```

```
// Server program: GossipServer.java
import java.io.*;
import java.net.*;

public class GossipServer
{

  public static void main(String[] args) throws Exception
  {


     ServerSocket sersock = new ServerSocket(3000);
     System.out.println("Server  ready for chatting");
     Socket sock = sersock.accept( );


                        // reading from keyboard (keyRead object)

     BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

                    // sending to client (pwrite object)

     OutputStream ostream = sock.getOutputStream();
     PrintWriter pwrite = new PrintWriter(ostream, true);


                         // receiving from server ( receiveRead  object)
     InputStream istream = sock.getInputStream();
     BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

     String receiveMessage, sendMessage;

     while(true)
     {

       if((receiveMessage = receiveRead.readLine()) != null)
      {

          System.out.println(receiveMessage);

      }

       sendMessage = keyRead.readLine();
       pwrite.println(sendMessage);
       pwrite.flush();

     }
    }
}
```

Practical No: 6. Write a program to retrieve date and time using TCP.

```
// Date Client

import java.io.*;
import java.net.*;

class DateClient
{
   publicstaticvoid main(String args[]) throws Exception
   {
      Socket soc=new Socket(InetAddress.getLocalHost(),5217);
      BufferedReader in=new BufferedReader(
            new InputStreamReader(
                  soc.getInputStream()
                  )
              );

      System.out.println(in.readLine());
   }
}
// Date Server

import java.net.*;
import java.io.*;
import java.util.*;

class DateServer
{
   publicstaticvoid main(String args[]) throws Exception
   {
      ServerSocket s=new ServerSocket(5217);

      while(true)
      {
         System.out.println("Waiting For Connection ...");
         Socket soc=s.accept();
         DataOutputStream out=new DataOutputStream(soc.getOutputStream());
         out.writeBytes("Server Date" + (new Date()).toString() + "\n");
         out.close();
         soc.close();
      }

   }
}
```

Practical No : 7. Write a program to retrieve date and time using UDP.

Server program

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

int main()
{
int sfd,r,bi,port;
char buff[1024];
struct sockaddr_in servaddr,cliaddr;
socklen_t clilen;
sfd=socket(AF_INET,SOCK_DGRAM,0);
if(sfd==-1)
{
perror("Socket");
return 0;
}

printf("\n Enter the port no:");
scanf("%d",&port);
printf("The port no is:%d\n",port);
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(port);
servaddr.sin_addr.s_addr = INADDR_ANY;

bi=bind(sfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
if(bi==-1)
{
perror("Bind()");
return 0;
}

clilen = sizeof(cliaddr);
r=recvfrom(sfd,buff,sizeof(buff),0,(struct sockaddr*)&cliaddr,&clilen);
buff[r]=0;
time_t ticks;
ticks = time(NULL);
snprintf(buff,sizeof(buff),"%24s\r\n",ctime(&ticks));
sendto(sfd,buff,sizeof(buff),0,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
exit(0);
return 0;
}
```

Client program

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

int main()
{
int listenfd,port,r;
char buff[1024];
struct sockaddr_in servaddr,cliaddr;
socklen_t servlen;
listenfd = socket(AF_INET,SOCK_DGRAM,0);
if(listenfd==-1)
{
perror("Socket");
return 0;
}

printf("\n Enter the port no:");
scanf("%d",&port);
printf("The port no is:%d",port);
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(port);
servaddr.sin_addr.s_addr = INADDR_ANY;
sendto(listenfd,buff,sizeof(buff),0,(struct sockaddr*)&servaddr,sizeof(servaddr));
r=recvfrom(listenfd,buff,sizeof(buff),0,(struct sockaddr*)&servaddr,&servlen);
buff[r]=0;
printf("\n The time received from the server:%s\n",buff);
exit(0);
return 0;
}
```

Practical No: 8. Write a client and server program to implement file transfer.

Program

File Server :

```
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class FileServer
{
 public static void main(String[] args) throws Exception
{
 //Initialize Sockets
 ServerSocket ssock = new ServerSocket(5000);
 Socket socket = ssock.accept();
 //The InetAddress specification
 InetAddress IA = InetAddress.getByName("localhost");

 //Specify the file
 File file = new File("e:\\Bookmarks.html");
 FileInputStream fis = new FileInputStream(file);
 BufferedInputStream bis = new BufferedInputStream(fis);
 //Get socket's output stream
 OutputStream os = socket.getOutputStream();
 //Read File Contents into contents array
 byte[] contents;
 long fileLength = file.length();
 long current = 0;
 long start = System.nanoTime();
 while(current!=fileLength){
 int size = 10000;
 if(fileLength - current >= size)
 current += size;
 else{
 size = (int)(fileLength - current);
 current = fileLength;
 }
 contents = new byte[size];
 bis.read(contents, 0, size);
 os.write(contents);
 System.out.print("Sending file ... "+(current*100)/fileLength+"% complete!");
 }
 os.flush();
 //File transfer done. Close the socket connection!
 socket.close();
 ssock.close();
 System.out.println("File sent succesfully!");
 } }
```

File Client

```
import java.io.BufferedOutputStream
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.Socket;
public class FileClient {
 public static void main(String[] args) throws Exception{
 //Initialize socket
 Socket socket = new Socket(InetAddress.getByName("localhost"), 5000);
 byte[] contents = new byte[10000];
 //Initialize the FileOutputStream to the output file's full path.
 FileOutputStream fos = new FileOutputStream("e:\\Bookmarks1.html");
 BufferedOutputStream bos = new BufferedOutputStream(fos);
 InputStream is = socket.getInputStream();
 //No of bytes read in one read() call
 int bytesRead = 0;
 while((bytesRead=is.read(contents))!=-1)
 bos.write(contents, 0, bytesRead);
 bos.flush();
 socket.close();
 System.out.println("File saved successfully!");
 }
}
```

Output

Server

```
E:\nwlab>java FileServer

Sending file ... 9% complete!
Sending file ... 19% complete!
Sending file ... 28% complete!
Sending file ... 38% complete!
Sending file ... 47% complete!
Sending file ... 57% complete!
Sending file ... 66% complete!
Sending file ... 76% complete!
Sending file ... 86% complete!
Sending file ... 95% complete!
Sending file ... 100% complete!
File sent successfully!
E:\nwlab>client
E:\nwlab>java FileClient
File saved successfully!
```