```
/*
  Computer Networks Laboratory (Lab) 15CSL77
  8. Using TCP/IP sockets, write a client-server program to make client sending
       the file name and the server to send back the contents of the requested
       file if present.
*/
                                    // Server
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define PORT 8080

int main()
 {
   int serverFd, newSocket;    // file (socket) descriptor
   struct sockaddr_in address; // socket address as sockaddr_in structure
   int option = 1;             // enable boolean option in setsockopt
   int addrlen = sizeof(address);
   char fileName[256] = {'\0'}, buffer[1024] = {'\0'};
        // save file name and the content of file
   int fileDesc;               // file descriptor

   // Create socket file descriptor , collect socket descriptor , check for error
   serverFd = socket( AF_INET, SOCK_STREAM, 0);

   // set options of sockets, like attaching socket to the port 8080 and
   setsockopt ( serverFd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                &option, sizeof(option));                      // check for error

   address.sin_family = AF_INET; //bind except when active listening socket bound
   address.sin_addr.s_addr = INADDR_ANY;//accept connections to all IPs of machine
   address.sin_port = htons( PORT );//convert between host and network byte order

   // bind a name to a socket,                                    check for error
   bind ( serverFd, (struct sockaddr *)&address, sizeof(address));

   listen (serverFd, 3); // listen for connections on a socket,   check for error

   newSocket = accept( serverFd, (struct sockaddr *)&address,  // accept a
         (socklen_t*)&addrlen );// connection on socket serverFd, check for error

   read( newSocket , fileName, 256); // read file name sent by client to socket
   printf("Client Request for file named : %s\n", fileName );
   // assume file exists and open requested file , collect file descriptor
   fileDesc = open( fileName, O_RDONLY ); //                    check for error

   while( read( fileDesc , buffer, 1024) ) //read until read() returns zero bytes
     send( newSocket, buffer, strlen(buffer), 0); // send read content to client

   printf("File content sent\n");
   return 0;
}

/* Output
   g++ server.c -o server

   Assume server has file called abc.txt
   cat abc.txt
   lmn
   pqr
   xyz

   ./server
```

```
   Client Request for file named : abc.txt
   File content sent
*/
```

```c
/*
  Computer Networks Laboratory (Lab) 15CSL77
  8. Using TCP/IP sockets, write a client-server program to make client sending
       the file name and the server to send back the contents of the requested
       file if present.
*/
                                    // Client
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define PORT 8080

int main( )
 {
   struct sockaddr_in address;      // socket address as sockaddr_in structure
   int socketFd = 0;                // file (socket) descriptor
   struct sockaddr_in serv_addr;    // socket address as sockaddr_in structure
   char fileName[18] = "abc.txt";   // file name to be requested by client
   char buffer[1024] = {'\0'};     // save content of file received from server

   // Create socket file descriptor , collect socket descriptor , check for error
   socketFd = socket( AF_INET, SOCK_STREAM, 0) ;
   // Initialize sockaddr_in structure variable
   memset( &serv_addr, '0', sizeof(serv_addr));

   serv_addr.sin_family = AF_INET;  //bind except when listening socket bound
   serv_addr.sin_port = htons(PORT);//convert between host and network byte order

   // Convert IPv4 and IPv6 addresses from text to binary form ,     why 127.0.0.1
   inet_pton( AF_INET, "127.0.0.1", &serv_addr.sin_addr) ;  //   hint XAMPP, WAMPP
                                                            //    check for error
   // connect the socket referred to by the file descriptor socketFd to the
   //   address specified by serv_addr,                      i.e. client to server
   connect( socketFd, (struct sockaddr *)&serv_addr, sizeof(serv_addr) );
                                                            //    check for error
   send( socketFd, fileName, strlen(fileName), 0);// send file name to server
   printf("File name sent to server\n");

   printf("File content received from server : \n");

   while( read( socketFd , buffer, 1024) )// read until server sends file content
       printf("%s",buffer);               //  into buffer and print the buffer

   return 0;
}
/* Output
   g++ client.c -o client

   Assume client has requested for file called abc.txt from server

   ./client
   File name sent to server
   File content received from server :
   lmn
   pqr
   xyz
*/
```