Q.1)

Solution:-

Client Side:-

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>

int main()

{

 int clientSocket;

 char buffer[1024];

 struct sockaddr\_in serverAddr, clientAddr;

 socklen\_t addr\_size;

 /\*---- Create the socket. The three arguments are: ----\*/

 /\* 1) Internet domain 2) Stream socket 3) Dlefault protocol (TCP in this case) \*/

 clientSocket = socket(PF\_INET, SOCK\_STREAM, 0);

 /\*---- Configure settings of the server address struct ----\*/

 /\* Address family = Internet \*/

 serverAddr.sin\_family = AF\_INET;

 /\* Set port number, using htons function to use proper byte order \*/

 serverAddr.sin\_port = htons(16027);

 /\* Set IP address to localhost \*/

 serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

 /\* Set all bits of the padding field to 0 \*/

 memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

 /\*---- Connect the socket to the server using the address struct ----\*/

 connect(clientSocket, (struct sockaddr \*)&serverAddr, sizeof(serverAddr));

 /\* ----  Send something to server ---- \*/

 printf("Enter message to send to server = ");

 scanf("%s", buffer);

 send(clientSocket, buffer, 1024, 0);

 recv(clientSocket, buffer, 1024, 0);

 printf("\nData received from server: %s\n", buffer);

 close(clientSocket);

 return 0;

}

Server Side:-

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>

int main()

{

 int welcomeSocket, newSocket;

 char buffer[1024];

 struct sockaddr\_in serverAddr;

 struct sockaddr\_in serverStorage;

 socklen\_t addr\_size;

 /\*---- Create the socket. The three arguments are: ----\*/

 /\* 1) Internet domain 2) Stream socket 3) Default protocol (TCP in this case) \*/

 welcomeSocket = socket(PF\_INET, SOCK\_STREAM, 0);

 /\*---- Configure settings of the server address struct ----\*/

 /\* Address family = Internet \*/

 serverAddr.sin\_family = AF\_INET;

 /\* Set port number, using htons function to use proper byte order \*/

 serverAddr.sin\_port = htons(16027);

 /\* Set IP address to localhost \*/

 serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

 /\* Set all bits of the padding field to 0 \*/

 memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

 /\*---- Bind the address struct to the socket ----\*/

 bind(welcomeSocket, (struct sockaddr \*)&serverAddr, sizeof(serverAddr));

 /\*---- Listen on the socket, with 5 max connection requests queued ----\*/

 if (listen(welcomeSocket, 5) == 0)

    printf("Listening\n");

 else

    printf("Error\n");

 /\*---- Accept call creates a new socket for the incoming connection ----\*/

 while (1)

 {

    addr\_size = sizeof serverStorage;

    newSocket = accept(welcomeSocket, (struct sockaddr \*)&serverStorage, &addr\_size);

    /\*---- Identify clients like this. The following information of client are taken from client due to connect function ----\*/

    //Change of the following information at client side can not be done. However, padding field may be changed and that may be tried

    struct sockaddr\_in \*cliIP = (struct sockaddr\_in \*)&serverStorage;

    struct in\_addr ipAddr = cliIP->sin\_addr;

    char str[INET\_ADDRSTRLEN];

    inet\_ntop(AF\_INET, &ipAddr, str, INET\_ADDRSTRLEN);

    char \*ID = cliIP->sin\_zero;

    char str2[8];

    inet\_ntop(AF\_INET, &ID, str2, 8);

    printf("\nClient IP is: %s", str);

    printf("\nClient port is: %d", serverStorage.sin\_port);

    printf("\nClient padding characters are (should be blank): ");

    int i;

    for (i = 0; i < 8; i++)

    {

     printf("%c", serverStorage.sin\_zero[i]);

    }

    /\*---- A  one liner ----\*/

    printf("\nGot a client connection from IP, port: <%s, %d> (can be used for client identification)\n", inet\_ntoa(serverStorage.sin\_addr), serverStorage.sin\_port);

    recv(newSocket, buffer, 1024, 0);

    printf("\nData received from client <%s, %d>: %s\n", inet\_ntoa(serverStorage.sin\_addr), serverStorage.sin\_port, buffer);

    strcpy(buffer, "Hello!\n");

    send(newSocket, buffer, 23, 0);

    close(newSocket);

 }

 return 0;

}

Output:-

CP connection,server receives a message from the client.If the message is “Hi”,server sends “Hello” to the client.