pt.No.

Page No.
Date.

LAB ASSIGNMENT - 3.

1) Write a server & client program for multi connection.

Input server_multi.c

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

#define PORT 4950
#define BUFSIZE 1024

void send_to_all (int j, int i, int sockfd, int nbytes_recvd,
                  char *recv_buf, fd_set *master).
{
    if (FD_ISSET (j, master)) {
        if (j != sockfd && j != i) {
            if (send (j, recv_buf, nbytes_recvd, 0) == -1)
            {
                perror ("send");
            }
        }
    }
}
```

Teacher's Signature .....

Page No.
Date.

```
void send_recv(int i, fd_set *master, int sockfd, int fdmax)
{
    int nbytes_recvd, j;
    char recv_buf[BUFSIZE], buf[BUFSIZE];
    if ((nbytes_recvd = recv(i, recv_buf, BUFSIZE, 0)) <= 0) {
        if (nbytes_recvd == 0) {
            printf("socket %d hung up\n", i);
        } else {
            perror("recv");
        }
        close(i);
        FD_CLR(i, master);
    } else {
        for (j = 0; j <= fdmax; j++) {
            send_to_all(j, i, sockfd, nbytes_recvd, recv_buf, master);
        }
    }
}

void connection_accept(fd_set *master, int *fdmax, int sockfd,
                       struct sockaddr_in *client_addr)
{
    socklen_t addrlen;
    int newsockfd;
    addrlen = sizeof(struct sockaddr_in);
    if((newsockfd = accept(sockfd, (struct sockaddr *)
                           client_addr, &addrlen)) == -1)
```

Teacher's Signature .....

```
{
    perror("accept");
    exit(1);
} else {
    FD_SET(newsockfd, master);
        if(newsockfd > *fdmax) {
            *fdmax = newsockfd;
        }

printf("New connection from %s on port %d \n", inet_ntoa(client_addr->
                                                    sin_addr),
                                                ntohs(client_addr->
                                                    sin_port));

void connect_request(int *sockfd, struct sockaddr_in *my_addr)
{
    int yes = 1;
    if(*sockfd = socket(AF_INET, SOCK_STREM)) == -1) {
        perror("Socket");
        exit(1);
    }

    my_addr -> sin_family = AF_INET;
    my_addr -> sin_port = htons(4950);
    my_addr -> sin_port.s_addr = INADDR_ANY;
    memset(my_addr -> sin_zero, '6', sizeof myaddr -> sin_zero);
```

```
if(setsockopt(*sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) == -1){
        perror("setsockopt");
        exit(1);
    }
if(bind(*sockfd, (struct sockaddr*) my_addr, sizeof(struct sockaddr))
                                                    == -1) {
        perror("Unable to bind");
        exit(1);
    }
if(listen(*sockfd, 10) == -1) {
        perror("listen");
        exit(1);
    }
    printf("\nTCPServer Waiting for client on port 4950\n");
    fflush(stdout);
}
int main() {
    fd_set master;
    fd_set read_fds;
    int fdmax, i;
    int sockfd = 0;
    struct sockaddr_in my_addr, client_addr;
        FD_ZERO(&master);
        FD_ZERO(&read_fds)
        connect_request(&sockfd, &my_addr);
        FD_SET(sockfd, &master);
```

Teacher's Signature .....

Page No.
Date.

```
while (1) {
    read_fds = master;
    if( select (fdmax+1, &read_fds, NULL, NULL, NULL
                                                 ==-1) {
        perror ("select");
        exit (4);
    }
    for ( i =0;  i <= fdmax ; i++) {
        if( FD_ISSET ( i, &read_fds)) {
            if ( i == sockfd)
    Connection_accept(&master, &fdmax, sockfd, &client_addr);
            else
    Send_recv ( i, &master, sockfd, fdmax);
            }
        }
    }
    return 0;
}
```

Page No.
Date.

client_multi.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <errno.h>
#define BUFSIZE 1024
void send_recv(int i, int sockfd)
{
    char send_buf[BUFSIZE];
    char recv_buf[BUFSIZE];
    int nbyte_recvd;
if (i==0){
    fgets(send_buf, BUFSIZE, stdin);
    if (strcmp(send_buf, "quit\n")==0){
        exit(0);
    } else {
        Send(sockfd, send_buf, strlen(send_buf), 0);
} else {
    nbyte_recvd = recv(sockfd, recv_buf, BUFSIZE, 0);
    recv_buf[nbyte_recvd] = '\0';
    printf("%s\n", recv_buf);
    fflush(stdout);
}
}
```

Teacher's Signature .....

Expt.No.

Page No.
Date.

```
void connect_request (int *sockfd, struct sockaddr_in *server_addr
{
    if ((*sockfd = socket (AF_INET, SOCK_STREAM, 0)) == -1) {
    perror ("Socket");
      exit (1);
    }

    Server_addr -> sin_family = AF_INET;
    Server_addr -> sin_port = htons (4950);
    server_addr -> sin_addr.s_addr = inet_addr ("127.0.0.1");
   memset (server_addr -> sin_zero, '6', sizeof server_addr -> sin_zero

if (connect (*sockfd, (struct sockaddr *) server_addr, sizeof (
                                        struct sockaddr)) == -1)
        {
          perror ("connect");
           exit (1);
        }
int main () {
    int sockfd, fdmax, i;
        struct sockaddr_in server_addr;
          fd_set master;
          fd_set read_fds;
    connect_request (& sockfd, & server_addr);
          FD_ZERO (& master);
            FD_ZERO (& read_fds);
            FD_SET (0, & master);
```

OXFORD

Teacher's Signature .....

No.

Page No.
Date.

```
FD_SET(sockfd, &master);
fdmax = sockfd;

while(1) {
    read_fds = master;
    if (select(fdmax+1, &read_fds, NULL, NULL, NULL)==-1)
    {
        perror("select");
        exit(4);
    }
for(i=0; i<fdmax; i++)
    if (FD_ISSET(i, &read_fds))
        send_recv(i, sockfd);
    }
printf("client-quited");
close(sockfd);
return 0;
}
```

Teacher's Signature .....

Output

New connection from Client on port 5500

Client: Hello!

Server: Hi!

Client: Computer.

Server: Networks.

Client: Lab.

Server: exit

Client Quit.