Certificate

Name: M . C . SOHAN Class:

Roll No: Exam No:

Institution R . V . College of Engineering

This is certified to be the bonafide work of the student in the Network Programming & Security *Laboratory during the academic year 2020 / 2021 .*

No. of practicals certified _____ *out of* _____ *in the subject of* Network Programming & Security.

.....
Teacher In-charge

.....
Examiner's Signature

.....
Principal

Date: .....

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

Output:

SERVER

```
$ ./filesrv
  Listen called
   Accepted
    text
    file sent
    Accepted
     notavailable.txt
```

CLIENT

```
$ ./filecli 127.0.0.1 text
   socket created : 3
   file text requested
    The contents of this file will be displayed
   l1
   l2
   End of file - there is a new line after this

$ ./filecli 127.0.0.1 notavailable.txt
    socket created : 3
    File notavailable.txt requested
    Unable to open the requested file.
```

Date

Expt. No. 1. Page No. 1

1. Implement a client and server communication using socket programming.

* File server

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

#define BUFSIZE 1024
#define PORT 5000
#define INADDR_ANY
#define BACKLOG 5

void servfunc (int sockfd);
```

// continued in the next page.

Teacher's Signature : _____

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M.C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 1

Title Client-server

```
int main()          //cout...
{
        struct sockaddr_in serv;
        int listenfd, acceptfd, fval;
        bzero(&serv, sizeof(serv));
        serv.sin_family = AF_INET
        serv.sin_port = htons(PORT)
        serv.sin_addr.s_addr = htonl(IP);

        // creating a concurrent server
        listenfd = socket(AF_INET, SOCK_STREAM, 0);
        if(listenfd < 0)
        {
                printf("Unable to create socket");
                exit(0);
        {
        // bind
if( ~~listen~~ bind( listenfd, (struct sockaddr*) &serv,
                                sizeof(serv)) == -1) {
```

Signature of
Teacher incharge

```
        printf("Bind error");
        exit(0);
}
if (listen(listenfd, BACKLOG) < 0)
  {
        printf("Unable to call listen");
          exit(0);
  }
printf("Listen called");
 while (1)
 {
        acceptfd = accept(listenfd, NULL, NULL)
        if (acceptfd == -1)
         {
            printf("Error accepting");
            exit(0);
         }
        printf("Accepted");
        fval = fork();
        if (fval == 0)
        {
              close(listenfd);
              // inside child
              serverfunc(acceptfd);
              close(acceptfd);
               exit(0);
                                    }
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN

Dept./Lab _____ Class _____ Expt./No. _____

Title _____

```
        close (acclphfd);
    }
    return 0;
}
void servfunc (int sockfd)
{
    char buf[BUFSIZE];
    int cnt, fd;
    cnt = recv (sockfd, buf, BUFSIZE, 0);
    if (cnt<0)
    {   printf("error");
        exit(0);
    }
    else if (cnt==0)
    {   printf ("No request");
        exit(0);
    }
```

Signature of
Teacher incharge

```
// write the name of the file
// open the file
// read the contents & send them to the cl
write (1, buf, cnt);
printf ("\n");
fd = open (buf, O_RDONLY);
if (fd == -1)
{
    sprintf (buf, BUFSIZE, "Unable
            to open requested file");
    send (sockfd, buf, strlen(buf),0
    return 0;
}
// send the file
bzero (buf, BUFSIZE);
while ( cnt = read ( fd, buf, BUFSIZE))
{
        send (sockfd, buf, cnt, 0);
        bzero (buf, BUFSIZE);
}
printf ("file sent\n");
close(fd);
} // end.
```

Date .....

Expt. No. 1 Page No. 2

File client

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

#define BUFSIZE 1024
#define PORT 5500
#define IP "127.0.0.1"

int main (int argc, char *argv[])
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
        if (argc != 3)
        {
            printf ("Format : ./a.out <IP> <file>\n");
            exit(0);
        }
        bzero (&serv, sizeof (serv));
        serv.sin_family = AF_INET;
        serv.sin_port = htons (PORT); // cont...
```

Teacher's Signature : _____

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M.C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 4

Title client-server (client cont...)

```
if (inet_pton(AF_INET, IP, (void *)&serv.sin_addr)
        == 0)
{
    printf("Invalid address");
    exit(0);
}
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
{
    printf("Cant create socket");
    exit(0);
}
printf("File %s requested \n", argv[2]);
// send(sockfd, argv[2]
if (connect(sockfd, (struct sockaddr *)&serv,
        sizeof(serv)) == -1)
{
    printf("Cannot connect");
    exit(0);
}
```

Signature of
Teacher incharge

```
    send( sockfd, argv[2], strlen(argv[2]), 0);
    while( cnt = recv( sockfd, buf, BUFSIZE, 0) )
        write (1, buf, cnt);
    close (sockfd);
    return 0;
} // end of client code
```

Text file that was used:

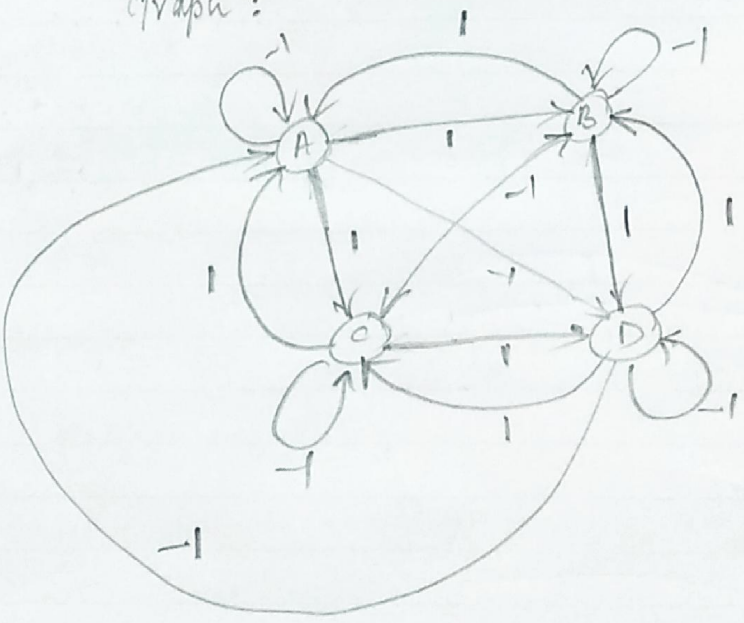
The contents of this file will be displayed.
l_1
l_2
End of file - there is a newline after this.

Output:

```
$ ./dv
Enter the no. of vertices : 4
Enter the adjacency matrix
-1  1  1  -1
 1 -1 -1   1
 1 -1 -1   1
-1  1  1  -1
Negative edge
```

Execution example for a graph with negative edge weights

Graph :



Another execution instance in the next page.

2. Write a program to implement distance vector routing protocol for a simple topology of routers.

```
#include <stdio.h>
#include <stdlib.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 ;
                    }
        }
        for ( u=0 ; u<n ; u++) {
            for( v=0 ; v<n ; v++) {
                if(d[v] > d[u] + A[u][v] ) {
                    printf ("Negative edge ");
                    exit(0);
                }
            }
        }
}  // end of function
```

// cont.

OUTPUT

```
$./dv
Enter no. of vertices : 4
Enter Adjacency matrix :
 999  7   2  999
   1  999 999  5
   1  999 999  1
 999  2   1  999
```

999
Here indicates the absence of an edge

Router 0

0 cost 0

1 ← 3 ← 2 ← 0 cost 5

2 ← 0 cost 2

3 ← 2 ← 0 cost 3

Router 1

0 ← 1 cost 1

1 cost 0

2 ← 0 ← 1 cost 3

3 ← 2 ← 0 ← 1 cost 4

Router 2

0 ← 2 cost 1

1 ← 3 ← 2 cost 3

2 ← cost 0

3 ← 2 cost 1

Router 3

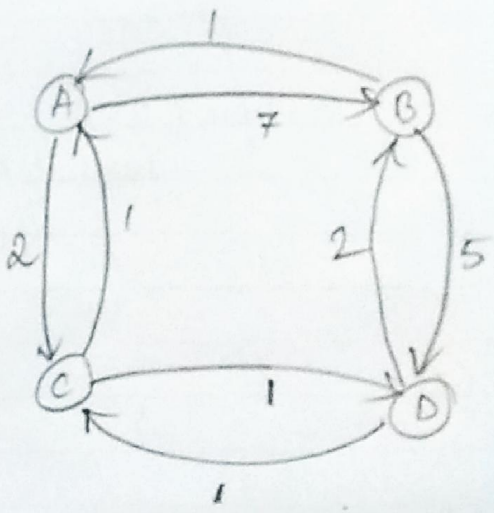
0 ← 2 ← 3 cost 2

1 ← 3 cost 2

2 ← 3 cost 1

3 cost 0

Graph



```
int main()
{
    printf("Enter the no. of vertices : "); scanf("%d", &n);
    printf("Enter adjacency matrix\n");
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            scanf("%d", &A[i][j]);
    int source=0;
    for(source=0; source<n; source++)
    {
        for(int i=0; i<n; i++){
            d[i] = 999;
            p[i] = -1;      {
        d[source] = 0;

        printf("Router %d\n", source);

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

Teacher's Signature : _____

Output (checksum) :

MODIFIED

```
$ ./ checksum
4500  0073  0000  4000  4011  b861  c0a8 0001
 c0a8   00C7
 Computed checksum : b861
 Enter 0 to modify, anything else to not modify
 0
 Invalid checksum
```

UNMODIFIED

```
$ ./ checksum
4500 0073 0000 4000 4011 b861 c0a8 0001
 c0a8   00c7                          (entered)
 Computed checksum : b861               1
 Valid checksum
 $
```

3. Write programs to implement error detection and correction concept using Checksum and Hamming code.

```
// Checksum
#include <stdio.h>
#include <stdlib.h>

unsigned int checksumGenerate (uint16_t * ipHeader)
{
    unsigned int sum = 0;
    ipHeader[5] = 0;
    for(int i=0; i<10; i++)
    {
        sum += ipHeader[i];
        while ((sum >> 16))
            sum = (sum & 0xFFFF) + 1u;
    }
    ipHeader[5] = (sum & 0xFFFF);
    ipHeader[5] = ~ipHeader[5];
    return ipHeader[5];
}

int verifyChecksum (uint16_t * ipHeader)
{
    unsigned int sum = 0u;

    //continued.
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN.

Dept./Lab _____ Class _____ Expt./No. _____

Title Checksum

```
for(int i=0; i<10; i++)
{
    sum += ipHeader[i];
    while (sum>>16)
        sum = (sum & 0xFFFF) + 1u;
}
if ((~sum & 0xFFFF) == 0)
    return 1;
else return 0;
}
int main()
{
    uint16_t ipfield[10];
    unsigned int sum, modify;
    for(int i=0; i<10; i++)
        scanf("%x", (ipfield+i));
    sum = checksumGenerate(ipfield) & 0xffff;
```

Signature of
Teacher incharge

```
printf ("Computed checksum: %x", sum);

printf ("\n Enter 0 to not modify, anything else
        to modify");
scanf ("%u", modify);
if (u)
    ipfield[0] = 0x0011 ^ ipfield[0];
// this changed (toggled) the position values

if (verifyChecksum (ipfield))
    printf ("Valid checksum");
else
    printf ("Invalid checksum");

return 0;
} // end of program.
```

Date

Expt. No. 3 Page No. 7

```
// Hamming code

#include <stdio.h>

int main()
{
    int bit[8];
    int rcvbit[8];
    int s, s0, s1, s2, i;
    printf(" enter 4 bits of data \n");
    for (i=0; i<4; i++)
        scanf("%d", &bit[i]);
// printf("Data entered!");
// calculate the values for the various bit positions
    bit[6] = bit[0] ^ bit[2] ^ bit[4];
    bit[5] = bit[0] ^ bit[1] ^ bit[4];
    bit[3] = bit[0] ^ bit[1] ^ bit[2];

printf("Data Encoded : \n");
printf(" d3 d2 d1 r2 d0 r1 r0 \n");
for( i=0; i<7; i++ )
        printf("%d", bit[i]);

printf(" Input received bits : \n");
    for(i=0; i<7; i++)
        scanf("%d", &rcvbit[i]);

// cont..
```

Teacher's Signature : _____

Output (Hamming Code):

```
$ ./hammingcode
Enter 4 bit data:
1 0 0 0
Encoded bits:
d3 d2 d1 r2 d0 r1 r0
1  0  0  1  0  1  1
Input received bits
1 0 0 0 0 1 1
Received sequence
d3 d2 d1 r2 d0 r1 r0
1  0  0  0  0  1  1
Error at position 4
Corrected message: 1001011
```

One complete execution

```
$ ./hamming code
```

2nd execution input

input given: 0110
for received bits: 0110011

```
Encoded bits:
d3 d2 d1 r2 d0 r1 r0
0  1  1  0  0  1  1

No error Detected.
```

Second execution Output

```
// calculate syndrome
s0 = rcvbit[6] ^ rcvbit[4] ^ rcvbit[2] ^ rcvbit[0];
s1 = rcvbit[5] ^ rcvbit[4] ^ rcvbit[1] ^ rcvbit[0];
s2 = rcvbit[3] ^ rcvbit[2] ^ rcvbit[1] ^ rcvbit[0];
s = (s2<<2) + (s1<<1) + s0;
if (s>0)
    printf ("No error detected");
else {
    printf ("\n Recieved sequence \n");
    printf (" d3 d2 d1 r2 d0 r1 r0\n");
    for( i=0; i<7 ; i++)
        printf("%d", rcvbit[i]);
    printf ("Error in position :");
    printf ("%d", s);
    if ( rcvbit[7-s] ==0)
        rcvbit[7-s] = 1;
    else
        rcvbit [7-s] = 0;
    for( i=0; i<7; i++)
        printf ("%d ", rcvbit[i]);

    }
printf ("\n");
return 0;
}   // end of program.
```

Output:

```
$ ./sender
^C
$
```

sender is executed, after a while it is aborted.

```
$ ./listener
RVCE-CSE
RVCE-CSE
RVCE-CSE
^C
$
```

The listener is started, allowed to receive messages for a while & then stopped. The messages are received with a 1s interval.

```
$./listner
RVCE-CSE
RVCE-CSE
RVCE-CSE
RVCE-CSE
RVCE-CSE
RVCE-CSE
RVCE-CSE
^C
$
```

The second listener is allowed to run simultaneously.

It too receives messages from the sender at an interval of 1s.

4. Implement a simple multicast routing mechanism.

/* Listener */

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.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()
{
    struct sockaddr_in addr;
    //int fd
    int fd, nbytes, addrlen;
    struct ip_mreq mreq;
    char msgbuf[MSGBUFSIZE];
    uint yes = 1;
    if((fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        printf("socket error");
        exit(0);
    }
```

//cont...

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 4

Title listener - multicast.

```
// modify the original socket.
if (setsockopt (fd, SOL_SOCKET, SO_REUSEADDR, &yes,
                sizeof(yes)) < 0)
{
    printf ("Error"); exit(0); }

// set the dubination address
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)
  { printf ("Bind error"); exit(0); }

mreq.imr_multiaddr.s_addr = inet_addr (HELLO_GROUP);
mreq.imr_interface.s_addr = htonl (INADDR_ANY);
```

Signature of
Teacher incharge

```
if (setsockopt( fd, IPPROTO_IP, IP_ADD_MEMBERSH
                & mreq, sizeof(mreq)) <0 )
{   printf ("sockopt error"); exit(0); }

  // read-print loop.

while(1)
{
    addrlen = sizeof (addr);
    if ( (nbytes = recvfrom( fd, msgbuf, MSGBU
                ,0, (struct sockaddr *)&addr,
                &addrlen ))<0 )
        printf("Error in recvfrom");
    puts(msgbuf);
}
return 0;
} // end of listener.
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN.

Dept./Lab _____ Class _____ Expt./No. 4

Title Sender

```
// sender - multicast.

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

#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"

int main()
{
    struct sockaddr_in addr;
    int fd, cnt;
    struct ip_mreq;
    char *msg = "RVCE-CSE";
```

Signature of Teacher incharge

Expt. No. 4

```
if ((fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        printf("socket error");
        exit(0);
}
memset(&addr, 0, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_addr.saddr = inet_addr(HELLO_GROU
addr.sin_port = htons(HELLO_PORT);
while(1)
{
    if(sendto(fd, msg, sizeof(msg), 0,
            (struct sockaddr *)&addr, sizeof(add
            < 0) {
        printf("send error"); exit(0);}
    sleep(1);
}
return 0;
} //end of program.
```

//end.

Output:

$ ./server

Listening

Client 2 disconnected!

Client 1

$ ./client

Hello
Received: Hello
Hi
Received: Good morning
Do you have some work now?

Received: Yes, Will talk later
Bye then
Received: Bye
^C
$

Client 2

$ ./client

Hello
Received: Hi
Good morning
Received: Do you have some work now?
Yes, will talk later
Received: Bye then
Bye
^C
$

Date .....

Expt. No. 5 Page No. 11

5. Write a program to implement concurrent chat server that allows current logged-in users to communicate with the other.

// Server

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

#define BUFSIZE 1024
#define PORT 5000
#define IP INADDR_ANY
#define BACKLOG 5

void servfunc (int sockfd1, int sockfd2)
{
    char buf[BUFSIZE];
    int cnt;
    cnt = recv(sockfd2, buf, BUFSIZE, 0);
    if (cnt==0)
    {   printf("Client 2 disconnected"); exit(0);}
```

Teacher's Signature : _____

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 5

Title _____

```
write(1, buf, cnt);
printf("2\n");
bzero(buf, BUFSIZE);
while(cnt = recv(sockfd1, buf, BUFSIZE, 0))
{
    send(sockfd2, buf, cnt, 0);
    write(1, buf, cnt);
    printf("1\n");
    bzero(buf, BUFSIZE);
    if(cnt = read(sockfd2, buf, BUFSIZE))
        send(sockfd1, buf, cnt, 0);
    else {
        printf("Client 2 disconnected");
        close(sockfd1);
        close(sockfd2);
    }
```

Signature of
Teacher incharge

```
        write(1, buf, cnt);
        printf("2 \n");
        bzero(buf, BUFSIZE);
    }
    printf("client disconnected");
}
int main()
{
        struct sockaddr_in serv;
        int listenfd, acceptfd1, acceptfd2, fval;
        bzero(&serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_port = htons(PORT);
        serv.sin_addr.s_addr = htonl(IP);
        listenfd = socket(AF_INET, SOCK_STRE
                                    0);
        if (listenfd < 0)
          {   printf("unable to create socket"
              exit(0);
          }
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. _____

Title _____

```
if (bind (listenfd, (struct sockaddr *) &serv,
        sizeof(serv)) == -1)
  { printf("Bind error"); exit(0); }

listen ( listenfd , BACKLOG) ;
while(1)
{
    acceptfd1 = accept ( listenfd , NULL, NULL);
    (printf(" Accepted from client 1") );
   acceptfd2 = accept ( listenfd, NULL, NULL);
    if ( acceptfd1 == -1 || acceptfd2 == -1)
     {
        printf( "Error accepting");
          exit(0);
     }
    servfunc (acceptfd1 , acceptfd2);
}
return 0; } //end.
```

Signature of
Teacher incharge

Date

C

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 5

Title _____

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

#define BUFSIZE 1024
#define PORT 5000
#define IP "127.0.0.1"

int main()
{
    struct sockaddr_in serv;
    char buf[BUFSIZE];
    int sockfd, cnt;
```

Signature of
Teacher incharge

```
bzero (&serv, sizeof(serv));
serv.sin_family=AF_INET
serv.sin_port = htons(PORT);
if (inet_pton (AF_INET, (void*) &serv.sin_addr
                                    ==0)
{
    printf ("Error\n");
        exit(0);
}
sockfd = socket( AF_INET, SOCK_STREAM, 0)
if ( connect(sockfd, (struct sockaddr*)&serv
                sizeof (serv)) == -1)
{
    printf ("Unable to connect\n");
        error exit(0);
}
printf ("Enter messages:\n");
//cont.
```

```
while (1)
{
        if ( cnt = read (1, buf, BUFSIZE) )
        {
            printf (" \n sent %.s \n", buf);
            send ( sockfd, buf, cnt, 0);
        }
        cnt = recv( sockfd , buf , BUFSIZE, 0 );
        printf ( "recv: ");
        write (1, buf, cnt);
        printf ( "\n");
        bzero ( buf, BUFSIZE);
    }
    close (sockfd);
    return (0);
} // end.
```

Date

Expt. No. 6. Page No. 13

6. Implementation of concurrent & iterative echo server using connectionless and connection oriented socket system calls.

* A new header file is created which will be included in all the subprograms.

```
// Headers.h

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

#define BACKLOG 5
#define BUFSIZE 1024.
```

Teacher's Signature :

Output:

```
$ ./ concurrent serv
 Listen called
   Accepted
   Accepted
    Sent
    Sent
```

Server

```
$ ./ connclient  127.0.0.1
    Socket created : 3
    Enter message :  Typo occurred
     Typo occurred
  $.
```

Client 1

```
$ ./ connclient  127.0.0.1
     Socket Created : 3
      Enter message ;  Hi
       Hi
$
```

Client 2.

a) Concurrent TCP server - client

```
//server
#include "Headers.h";
#define PORT 5000
#define IP INADDR_ANY

void servfunc (int sockfd)
{
        char buf[BUFSIZE];
        int cnt;
        bzero (buf, BUFSIZE);
        cnt = recv (sockfd, buf, BUFSIZE, 0);
        if (cnt<0)
        {       exit(0); }
        else if (cnt>0)
        {
                send (sockfd, buf, cnt, 0);
                bzero (buf, BUFSIZE);
        }
        printf ("Sent\n");

}

int main ()
{
        struct sockaddr_in serv;
        int listenfd, acceptfd, fval;
        bzero (&serv, sizeof(serv));
```

//cont...

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6a

Title _____

```
// Server
    serv.sin_family = AF_INET;
    serv.sin_port = htons(PORT);
    serv.sin_addr.s_addr = htons(IP);
    listenfd=Socket (AF_INET, SOCK_STREAM, 0);
    if (bind (listenfd, (struct sockaddr *) &serv,
                    sizeof (serv)) == -1 ) {
        printf( "Error bind "); exit(0);
    }
    listen (listenfd, BACKLOG, 0 );
    printf ( "Listen called");
    while (1)
    {
        acceptfd = accept (listenfd, NULL, NULL);
        if (acceptfd == -1)
        { printf ("Error accepting");
          exit(0);
        } printf ( "Accepted");
```

Signature of
Teacher incharge

```
        fval = fork()
    6   if (fval == 0)
        {
            close(listenfd);
            servfunc(acceptfd);
            close(acceptfd);
            exit(0);
        }
        close(acceptfd);
    }
    return 0;
} //end of server.
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6a

Title Client

```
// Client
#include "Headers.h"
# define IP "127.0.0.1"
# define PORT 5000

int main()
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
        bzero(&serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_port = htons(PORT);
        if(inet_pton(AF_INET, IP, (void *)&serv.sin_
                                        addr) == 0)
        {   printf("Invalid addr \n");
            exit(0);
        }
        printf("socket created");
```

Signature of
Teacher incharge

```
if (connect(sockfd, (struct sockaddr *) &serv,
                    sizeof(serv)) == -1 )
  {
      printf(" Error connecting");
       exit(0);
  }
printf ("Enter Message:\n");
bzero(buf, BUFSIZE);
fgets( buf, BUFSIZE, stdin);
  send (sockfd, buf, strlen(buf),0);
    bzero(buf);
  cnt = recv( sockfd, buf, BUFSIZE, 0);
    if (cnt>0)
     { write(1, buf, cnt); }
       close (sockfd);
    return 0;
} // end of 6a.
```

Output:

```
$ ./ iterativeconnserv
  Listen called
   Accepted
   Accepted.
```

Server

```
$ ./ connclient 127.0.0.1
     Socket created :3
      Enter message: Good afternoon
      Good Afternoon
$
```

Client 1

```
$ ./ connclient 127.0.0.1
     Socket created : 3
      Enter message : Hello
       Hello.
```

Client 2

6.b) Iterative TCP server (echo server) - client.

```
// Server - connection oriented, iterative
#include "Headers.h"

#define PORT 5000
#define IP INADDR_ANY

void servfunc (int sockfd)
{
        char buf[BUFSIZE]
        int cnt;
        bzero( buf, BUFSIZE);
        cnt = recv( sockfd, buf, BUFSIZE, 0);
        if( cnt>0)
        {
                send (sockfd, buf, cnt, 0);
                bzero( buf, BUFSIZE);
        }
}

int main ()
{
        struct sockaddr_in srv;
        int listenfd, acceptfd, fval;
        bzero (&srv, sizeof(srv));
        srv.sin_family = AF_INET;
        srv.sin_port = htons( PORT);
        srv.sin_addr.s_addr = htonl (IP);
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6B

Title _____

```
//server

listenfd = ~~listen~~ socket (AF_INET, SOCK_STREAM, 0);
if ( bind (listenfd, (struct sockaddr*) &srv,
        sizeof (srv)) == -1)
{
    printf ( "Bind error");
    exit (0);
}
if (listen ( listenfd, BACKLOG) < 0)
{
    printf ( "Listen error");
    exit (0);
}
printf ( "Listen called");

// cont...
```

Signature of
Teacher incharge

```
// servfunc to be called.

while (1)
{
        acceptfd = accept (listenfd, NULL, NULL)
        if (acceptfd == -1)
        {
            printf ("Error accepting");
            exit(0);
        }
        printf ("Accepted");
        servfunc (acceptfd);
        close (acceptfd);
    }
    return 0;
}
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 8B

Title Client

```
// Client code - iterative, connection oriented.
#include "Headers.h"
#define PORT 5000
#define IP "127.0.0.1"

int main (int argc, char * argv[])
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;

        if (argc != 2)
        {
            printf ("Format : ./a.out <IP>\n");
            exit(0);
        }

        bzero( &serv, sizeof (serv));
        serv. sin_family=AF_INET;
        serv. sin_port = htons
                        (PORT);
```

Signature of
Teacher incharge

```
if ( inet_pton (AF_INET, argv[1], (void *) &servaddr.sin_
                addr ) == 0 )
{
        printf ("Invalid address");
        exit(0);
}
sockfd = socket (AF_INET, SOCK_STREAM, 0);
if (connect ( sockfd, (struct sockaddr *) &serv,
{           sizeof (serv) ) == -1).
        printf ("unable to connect");
        exit(0);
}
printf ("Enter message: \n");
bzero (buf, BUFSIZE);
fgets( buf, BUFSIZE, stdin );
send ( sockfd, buf, strlen(buf), 0);
bzero ( buf, BUFSIZE);
cnt = recv ( sockfd, buf, BUFSIZE, 0);
// cont..
```

```
// client cont...

if (cnt < 0)
  {
      printf ("Error");
  }
else if (cnt > 0)
  {
      write (1, buf, cnt);
  }
  close (sockfd);

  return 0;
}
// end of client for 6b.
```

Output :

Server

```
$ ./iterativeconnlessserv
Accepted
Sent
Accepted
Sent
```

Client 1

```
$ ./connlessclient 127.0.0.1
socket created : 3
Enter message : Helloworld
Helloworld.
$
```

Client 2.

```
$./connlessclient 127.0.0.1
socket created : 3
Enter message : No typo
No typo
```

6.c) UDP echo

```
// server
#include "Headers.h"
# define PORT 5000
# define IP INADDR_ANY

void servfunc ( int sockfd, struct sockaddr_in *client , int *len)
{
        char buf[BUFSIZE] ;
        int cnt;
        bzero ( buf, BUFSIZE);
        cnt = recvfrom ( sockfd, buf, BUFSIZE, 0 ,
            (struct sockaddr *) client , (socklen_t*)len );
        if ( cnt >0 )
                sendto ( sockfd, buf, cnt, 0 ,
                    (struct sockaddr *) client, *len );
        else
          {  printf (" Nothing received ");
             exit(0);
          }
         printf ( " sent \n" );
}

int main ()
{
        struct sockaddr_in serv;
        int sockfd, fval ;
        bzero( &serv, sizeof(serv) );
```

Teacher's Signature :

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 6

Title 6.c)

```
// cont... server
serv.sin_family = AF_INET;
serv.sin_port = htons(PORT);
serv.sin_addr.s_addr = htonl(IP);
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (bind(sockfd, (struct sockaddr*) &serv,
                sizeof(serv)) == -1)
{
    printf("Bind failed");
    exit(0);
}
struct sockaddr_in client;
int len;
while(1)
{
    len = sizeof(client);
    bzero(&client, len);
    servfunc(sockfd, &client, &len);
}
return 0;
} // end of server.
```

Signature of
Teacher incharge

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M.C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 6

Title 6. C

```
// Client
#include "Headers.h"
#define PORT 5000
#define IP "127.0.0.1"

int main (int argc, char * argv[])
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
        if (argc != 2)
        {
            printf ("Format: ./a.out <IP> \n");
            exit(0);
        }
        bzero( & serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_addr
        serv.sin_port = htons( PORT);
//cont...
```

Signature of
Teacher incharge

Expt. No. ----------

```
if ( inet_pton( AF_INET, IP, (void *) & serv.sin_addr
                == 0 )
{
        printf( "Invalid address \n");
        exit(0);
}
sockfd = socket( AF_INE, SOCK_DGRAM, 0);
printf (" Socket created ");
printf (" Enter the message\n");
bzero ( buf, BUFSIZE );
fgets( buf, BUFSIZE, stdin);
cnt = sendto( sockfd, buf, strlen(buf), 0,
      (struct sockaddr *) & serv, sizeof (serv))
if (cnt<0)
{
        printf ("Unable to send \n");
        exit(0);
}
bzero ( buf, BUFSIZE);
cnt = recvfrom ( sockfd, buf, BUFSIZE, 0,
                 NULL, NULL) ;
if (cnt>0)
  write ( 1, buf, cnt);
        close (sockfd);
        return 0;
}
//end.
```

Output :

$ ./ rcserver

Listen called

Accepted

ls

rm: cannot remove 'temp' : no such file / directory

ls > temp

Result sent

who > temp

Result sent

Accepted

pwd

pwd > temp

Result sent

server

Output

instance of client.

$ ./ rcclient 127.0.0.1

Enter command : who

Received :

Kali tty7 2020-11-17 03:36

rc

Only a part has been written.

$ ./ rcclient 127.0.0.1

Enter command : ls -l

Received :

total 48

-rwxr-xr-x 1 kali kali 17176 Nov 17 05:02 rcclient

-rw-r--r-- 1 kali kali 1616 Nov 17 04:44 rcclient.c

:

:

7. Implementation of remote command execution using socket system calls.

```
// Remote command server
#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <fcntl.h>

#define BUFSIZE 1024
#define PORT 5000
#define IP INADDR_ANY
#define BACKLOG 5

void servfunc(int sockfd)
{
        char buf[BUFSIZE];
        int cnt, fd;
        cnt = recv(sockfd, buf, BUFSIZE, 0);
        write(1, buf, cnt);
        buf[cnt-1] = '\0';
        system("rm temp");
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 7

Title _____

```
//7. server cont...
    strcat ( buf , " > temp");
    printf ( "%s\n", buf );
     system (buf);
     system ( "chmod a+r temp");
     printf ("\n");
     fd = open ( "temp" , O_RDONLY ) ;
     if ( fd == -1)
     {
          snprintf ( but, BUFSIZE, "Error\n");
          send ( sockfd, buf, strlen(buf), 0);
          return ;
     {
     bzero (buf, BUFSIZE);
     while ( cut = read ( fd, buf, BUFSIZE) )
     {
          send ( sockfd, buf, cut, 0);
          bzero ( buf, BUFSIZE);
     }
```

Signature of
Teacher incharge

```
        printf("Result sent\n");
        close(fd);
}
int main()
{
        struct sockaddr_in serv;
        int listenfd, acceptfd, fval;
        bzero(&serv, sizeof(serv));
        serv.sin_family = AF_INET;
        serv.sin_port = htons(PORT);
        serv.sin_addr.s_addr = htonl(IP);
        listenfd = socket(AF_INET, SOCK_STREAM, 0);
        if (bind(listenfd, (struct sockaddr*)&serv,
                sizeof(serv)) == -1)
        {
                printf("Error bind");
                error(0);
        }
        if (listen(listenfd, BACKLOG) < 0)
        {
                printf("Unable to call listen\n");
                exit(0);
        }
        printf("Listen called");
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. _____

Title _____

```
// cont
    while(1)
    {
        acceptfd = accept(listenfd, NULL, NULL);
        if (acceptfd == -1)
        {
            exit(0);
        }
        printf("Accepted\n");
        fval = fork();
        if( fval==0)
        {
            close (listenfd);
            servfunc (acceptfd);
            close( acceptfd);
            exit(0);
        }
        close(acceptfd);
    }
    return 0;
} // end.
```

Signature of
Teacher incharge

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name _____

Dept./Lab _____ Class _____ Expt./No. 7

Title _____

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

#define BUFSIZE 1024
#define PORT 5000
#define IP "127.0.0.1"
int main (int argc, char* argv[])
{
        struct sockaddr_in serv;
        char buf[BUFSIZE];
        int sockfd, cnt;
        if (argc != 2)
        { printf("Format: ./a.out <IP>"); exit(0); }
```

Signature of
Teacher incharge

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (connect(sockfd, (struct sockaddr *) &
            sizeof(serv)) == 1)
{
    printf("Unable to connect");
    exit(0);
}
printf("Enter command \n");
bzero(buf, BUFSIZE);
cnt = read(1, buf, BUFSIZE);
send(sockfd, buf, strlen(buf), 0);
bzero(buf, BUFSIZE);
printf("Recieved \n");
while(1)
{   while(cnt = recv(sockfd, buf, BUFSIZE,
            write(1, buf, cnt);
}
close(sockfd);
return 0;
} //end.
```

// Remote command client

→ Code on observation sheet.

8. Write a program to encrypt and decrypt the data using RSA and exchange key securely using Deffie-Hellman key exchange.

// both the programs use a common function which is separately compiled

```
// exponentiation.h
#include <stdio.h>
int exponentiation (int a, int x, int n);
```

```
// exponentiation.c
#include "exponentiation.h"

int exponentiation (int a, int x, int n)
{
    int dp[1024];
    dp[0] = 1;   dp[1] = a%n;
    for (int i = 2; i<x; i++)
    {
        dp[i] = (dp[i/2] * dp[i/2]) %n;
        if (i%2)  dp[i] = (dp[i] * dp[1])%n;
    }
    if (x>=0)
        return dp[x];

    return 0;
}
```

OUTPUT

```
$ gcc -o rsa rsa.c exponentiation.o

$ ./rsa

Format: ./a.out p q x d

$ ./rsa 11 13 23 47
Enter a string of not more than 9 english alphabets
HELLO
5                    // program prints the length
Encrypted message:
2 75 110 110 27
Decrypting the same message:
HELLO

$ ./rsa 11 13 23 47
Enter a string of not more than 9 english alphabets
morning
7
Encrypted msgae
99 41 4 89 79 80 103
Decrypting the same message:
morning.

$
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M.C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 8

Title RSA

```
// RSA Encryption - decryption
#include "exponentiation.h"
#include <stdlib.h>
#include <string.h>

int main (int argc, char * argv[])
{
    int p, q, n, a, x, d, len;
    if (argc != 5)
    {
        printf("Format: ./a.out p q x d");
        return 0;
    }
    char input[10];
    int arr[10];
    printf("Enter a string of not more than 9
            english alphabets\n");
```

Signature of
Teacher incharge

```
scanf("%s", input)
len = strlen(input);
printf("%d", len);
p = atoi(argv[1]);
q = atoi(argv[2]);
x = atoi[argv[3]);
d = atoi(argv[4]);
n = p*q;
printf("Encrypted message: \n");
for(int i=0; i<len; i++)
{
    a = input[i] - 'A';
    arr[i] = exponentiation(a, x, n);
    printf("%d", arr[i]);
}
printf("Decrypting the same encrypted
        message: \n");
for(int i=0; i<len; i++)
{
    printf("%c", (char)(exponentiation(
        arr[i], d, n) + 'A'));
}
return 0;
} // end of program.
```

// code on datasheet

RSA Encryption.

OUTPUT

```
$ gcc -o dhke dhke.c -lpowermation.o
$./dhke 23 9 4 3

  Ya = 6
  Yb = 16
  Key computed by A : 9
  Key computed by B : 9

$ ./dhke

 Format: ./a.out q alpha xa xb

$
$./dhke 11 2 9 4
  YA = 6
  Yb = 5
  Key computed at A : 9
  Key computed at B : 9

$
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN

Dept./Lab _____ Class _____ Expt./No. 8

Title Deffie Hellman key exchange

```
#include "exponentiation.h" // created file
#include <stdlib.h>

int main ( int argc, char* argv[])
{
        int q, alpha, xa, xb; // input
        int ya, yb, key ; //calculated
        if (argc != 5)
        {
            printf ("format: ./a.out q alpha xa xb");
            return 0;
        }
        q = atoi (argv[1]);
        alpha = atoi (argv[2]);
        xa = atoi (argv[3]);
        xb = atoi (argv[4]);

        //continued :
```

Signature of
Teacher incharge

```
ya = exponentiation (alpha, xa, q);
yb = exponentiation (alpha, xb, q);
printf ("\n ya = %d \n yb = %d \n", ya, yb);
printf ("\n Key computed at side A: %d",
                exponentiation (yb, xa, q));
printf ("\n Key computed at side B: %d",
                exponentiation (ya, xb, q));
printf ("END\n");
return 0;
} // end of program.
```

Pt. No. 8

Diffie Hellman key exchange

// Code on Datasheet.