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

```
$ ./ rcclient 127.0.0.1
 Enter command : who
  Received :
   kali        tty7     2020-11-17  03:36
  ^c

$ ./ 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
      :
```

instance of client.

Only a part has been written.

Date

Expt. No. 7 Page No. 20

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");
```

Teacher's Signature :

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 ( buf, BUFSIZE, "Error\n");
          send ( sockfd, buf, strlen(buf), 0);
          return ;
       }
       bzero (buf, BUFSIZE);
       while ( cnt = read ( fd, buf, BUFSIZE))
       {
            send ( sockfd, buf, cnt, 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*)&se
                    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.

Date

Dept./Lab

Title

// (

}

re

} // en

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 ops"); exit(0); }
```

Signature of
Teacher incharge

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if ( connect ( sockfd, (struct sockaddr *) &serv,
                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 observationsheet.