Certificate

Name: NACHIKET G KALLAPUR Class:

Roll No: Exam No:

Institution _____

This is certified to be the bonafide work of the student in the _____ Laboratory during the academic year 20 / 20 .

No. of practicals certified _____ out of _____ in the subject of _____

.....
Teacher In-charge

.....
Examiner's Signature

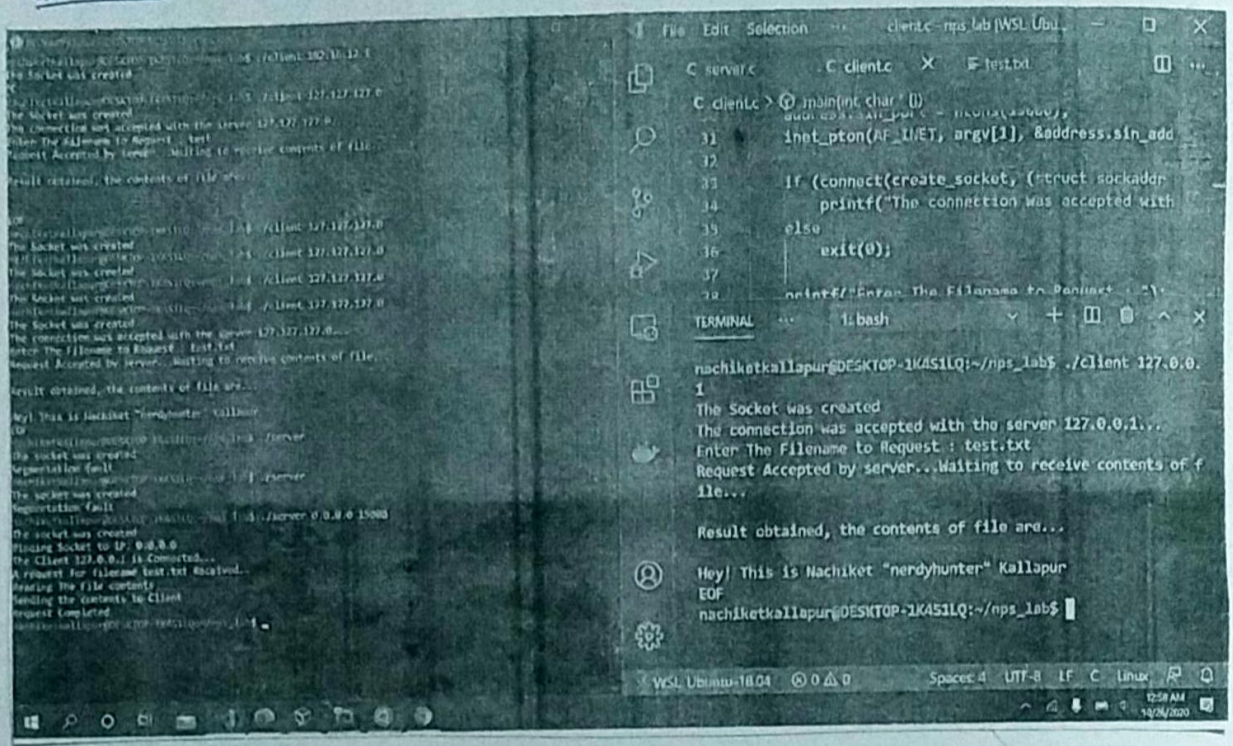
.....
Principal

Date: .....

Institution Rubber Stamp

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

Output (program)



Expt. No. _____ Page No. _____

1) Implement a client-server communication using socket programming

Server

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet.h>
#include <sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include <fcntl.h>
#include<arpa.inet.h>
#include <string.h>
int main(int argc, char*argv[]){
    int cont, create_socket, new_socket, addrlen, fd;
    int buffsize = 1024;
    char * buffer = malloc(buffsize);
    char fname[256];
    struct sockaddr_in address;

    if ((create_socket = socket(AF_INET, SOCK_STREAM,0)) >0)
        printf("Socket was created successfully");
    else exit(0);
    address.sin_family =AF_INET;
    inet_pton(AF_INET, argv[1], &address.sin_addr);
    address.sin_port = htons(atoi(argv[2]));
    if(bind (create_socket, (struct sockaddr*)&address, sizeof(address))==0)
        printf("Binding socket to IP : %s\n", inet_ntoa (address.sin_addr));
```

Teacher's Signature _____

Expt. No. _____ Page No. _____

```
else exit(0);
listen(create_socket,2);
addrlen = sizeof(struct sockaddr_in);

new_socket = accept(create_socket,(struct sockaddr*)&address,&addrlen);
if(new_socket >0){
   printf("The client is connected ... \n", inet_ntoa(address.sin_addr));
else exit(0);

recv(new_socket, fname,255,0);
printf(" A request for filename %s received ...\n", fname);
if((fd= open(fname, O_RDONLY)) <0){
   perror("File open failed "); exit(0);
}
while((cnt = read(fd, buffer, buffsize)) >0){
    printf("Reading File contents\n");
    printf(" Sending the contents to client \n");
    send(new_socket, buffer, cnt, 0);
}

printf("Request Complete\n");
close(new_socket);
return close(create_socket);
}
```

Teacher's Signature _____

Date

Expt. No.

Page No.

Client

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

int main(int argc, char *argv[]){
    int create_socket, buffsize=1024, cont;
    char *buffer = malloc(buffsize);
    char fname[256];
    struct sockaddr_in address;

    if((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
        printf("Socket was created\n");
    else exit(0);

    address.sin_family = AF_INET;
    address.sin_port = htons(15002);
    inet_pton(AF_INET, argv[1], &address.sin_addr);

    if(connect(create_socket, (struct sockaddr*)&address, sizeof(address)) = 0)
        printf("The connection was accepted with the server %s... \n", argv[1]);
    else exit(0);
```

Teacher's Signature

Expt. No. _____ Page No. _____

```
printf("Enter the filename to request : ");
scanf("%s", fname);
send(create_socket, fname, sizeof(fname),0);
printf("Request accepted by the server... Waiting to receive the
contents of the file...\n\n");
printf("Results obtained, the contents of the file are...\n\n");

while ((cont = recv(create_socket, buffer, buffsize,0)) >0)
    write(1, buffer, cont);

printf("\nEOF\n");
return close(create_socket);
```

Teacher's Signature _____

Output - Distance Vector

```
From Router 4 to Router 2
Cost: 7 | Path: 2 <- 5 <- 4

From Router 4 to Router 3
Cost: 4 | Path: 3 <- 4

From Router 4 to Router 4
Cost: 0 | Path: 4

From Router 4 to Router 5
Cost: 2 | Path: 5 <- 4

From Router 5 to Router 1
Cost: 10 | Path: 1 <- 2 <- 5

From Router 5 to Router 2
Cost: 9 | Path: 2 <- 5

From Router 5 to Router 3
Cost: 6 | Path: 3 <- 4 <- 5

From Router 5 to Router 4
Cost: 2 | Path: 4 <- 5

From Router 5 to Router 5
Cost: 0 | Path: 5
nachiketkallapur@DESKTOP-1K6S1LQ:~/NPS_Lab/program2$
```

Date

Expt. No. Page No.

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

→ Code

```
#include<stdio.h>
int A[10][10], n, d[10], p[10];
int bellman_ford(){
  int i,u,v;
  for(int i=0; i<n; i++){
    for(int 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("Detected negative edge");
        return -1;
      } } }
  return 0;
}

int main(){
  printf("N: "); scanf("%d", &n);
  printf("Matrix:\n");
  int source=0, destination=0;
  for(int i=0; i<n; i++) for(int j=0; j<n; j++) scanf("%d", &A[i][j]);
```

Teacher's Signature

Date

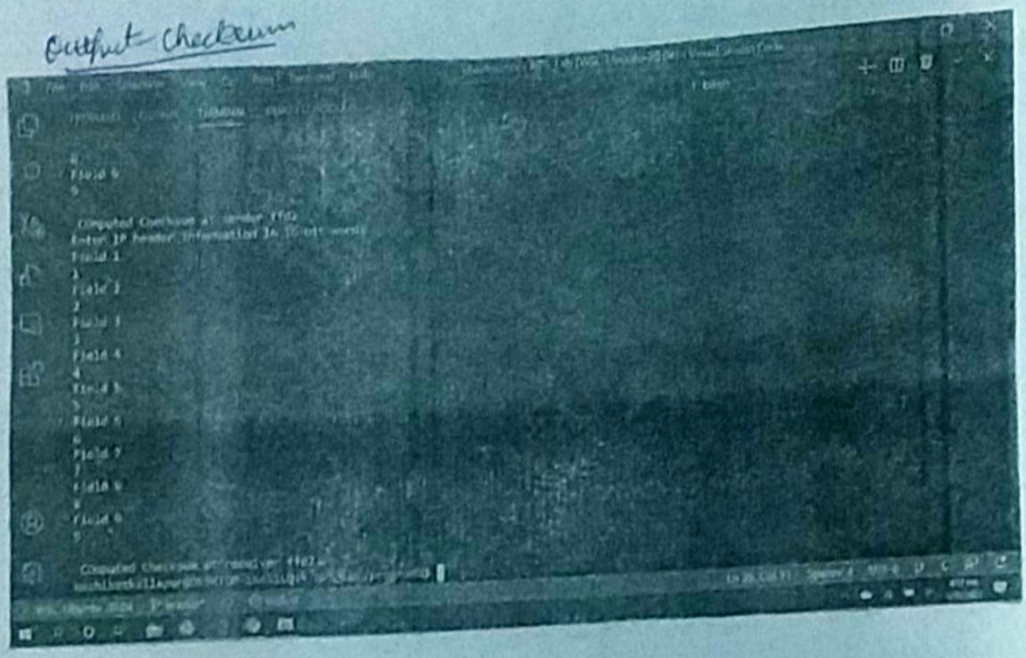
Expt. No.

Page No.

```
for(source=0; source<n; source++){
    for(destination=0; destination<n; destination++){
        for(int i=0; i<n; i++){ d[i]=999, p[i]=-1; }
        d[source]=0;
        int valid = bellman-ford();
        if(valid==-1){ printf("Negative edge detected \n"); return -1; }
        printf("\From Router %d to Router %d \n", source+1, destination+1);
        printf(" cost : %2d | Path ", d[destination]);
        if(destination != source){
          int j=destination;
          while(p[j]!=-1){
              printf("%d <- ", j+1);
              j=p[j];
          }
        }
        printf("%d\n", source+1);
      }
    }
}
```

Teacher's Signature

Output checksum



Date

Expt. No.

Page No.

3) write a program to implement error detection and correction using checksum and hamming code

→ checksum

```
#include <stdio.h>
unsigned fields [10];
unsigned short checksum(){
  int i;
  int sum=0;
  printf("Enter IP header information in 16 bit words\n");
  for(i=0; i<9; i++){
      printf("Field %d\n", i+1);
      scanf("%x", &fields[i]);
      sum = sum + (unsigned short) fields[i];
      while(sum >>16)
      ~~sum~~ sum = (sum & 0xFFF) + (sum>>16);
    }
  ~~sum~~
  sum = ~sum;
  return (unsigned short) sum;
}

int main(){
  unsigned short result1, result2;
  result1 = checksum();
  printf("\n Computed checksum at sender %x\n", result1);
```

Teacher's Signature

Date

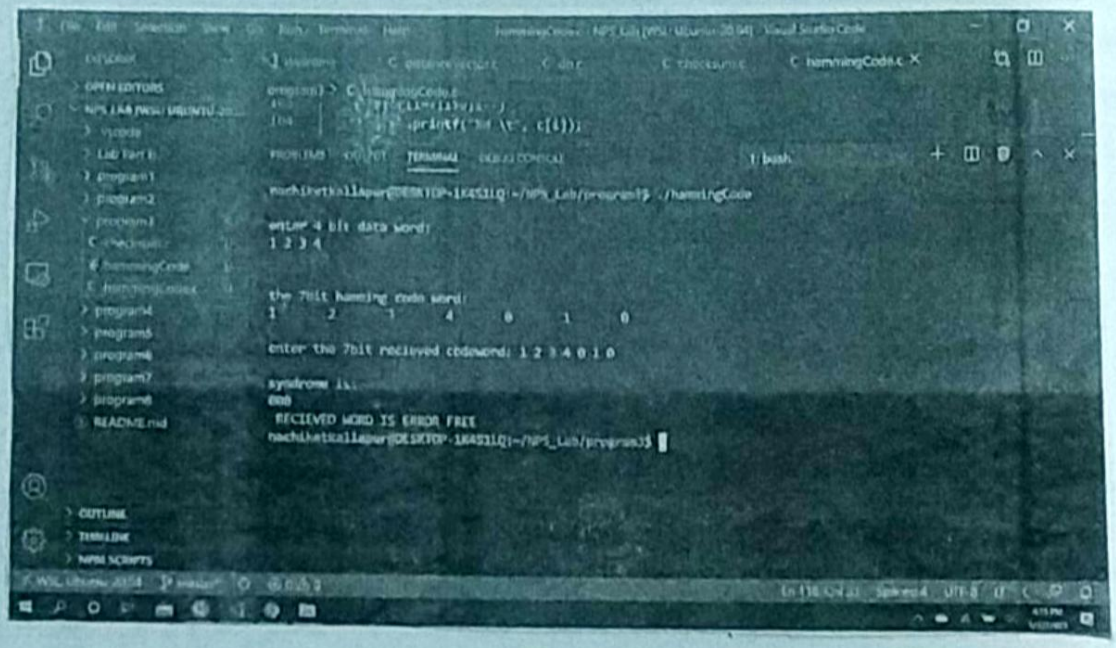
Expt. No.

Page No.

```
//Receiver
result2=checksum();
printf("\n computed checksum at receiver %x\n", result2);
if (result1 == result2)
    printf("No error");
else
    printf("Error in data received");
}
```

Teacher's Signature

output - Hamming code



Expt. No. _____ Page No. _____

Hamming code:

```
#include<stdio.h>
void main(){
  int data[10];
  int dataatrec[10], c, c1, c2, c3, i;
  printf("Sender: ");
  scanf("%d %d %d %d", &data[0], &data[1], &data[2], &data[4]);
  data[6] = data[0]^data[2]^data[4];
  data[5] = data[0]^data[1]^data[4];
  data[3] = data[0]^data[1]^data[2];
  printf("Encoded data: ");
  for(i=0; i<7; i++) printf("%d", &data[i]);
  printf("Receiver: ");
  for(i=0; i<7; i++) scanf("%d", &dataatrec[i]);
  c1 = dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
  c2 = dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
  c3 = dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
  printf("\n Syndrome bits: %d %d %d", c1, c2, c3);
  if (c==0) printf("No error while transmission of data");
  else {
    printf("\n Error position: %d ", c);
    printf("\n Data sent: ");
    for(i=0; i<7; i++) printf("%d", data[i]);
    printf("\n Data Received: ");
    for(i=0; i<7; i++) printf("%d", dataatrec[i]);
    printf("\n Corrected Message is: ");
    if(dataatrec[7-c]==0) dataatrec[7-c]=1;
    else dataatrec[7-c]=0;
```

Teacher's Signature _____

Date

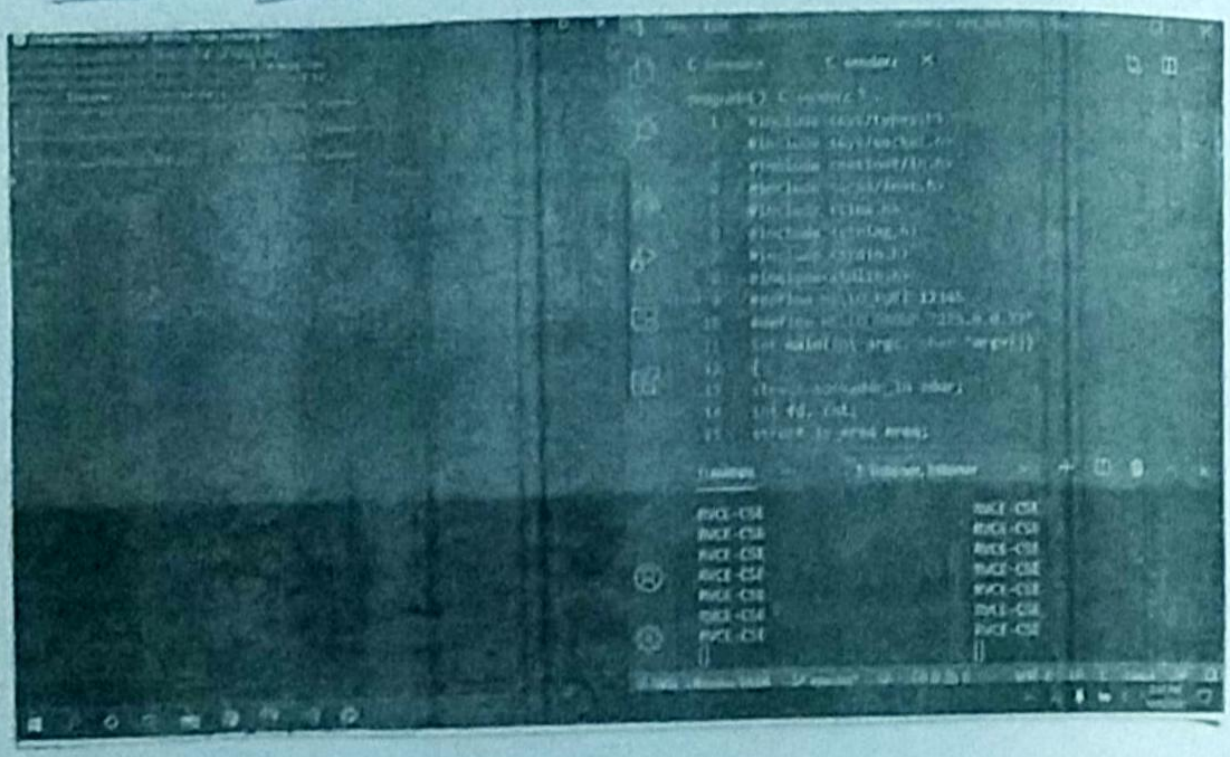
Expt. No.

Page No.

```
for (i=0; i<9; i++) {
    printf ("%d", dataatrec[i]);
  }
}
}
```

Teacher's Signature

output - Multicast program



Date

Expt. No.

Page No.

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 "225.0.0.37"
#define MSGBUFSIZE 1024000
int main(int argc, char*argv[]){
    struct sockaddr_in addr;
    int fd; int nbytes, addrlen, rl;
    struct ip_mreq mreq;
    char msgbuf[MSGBUFSIZE];
    u_int yes=1;
    int if((fd=socket(AF_INET, SOCK_DGRAM, 0))<0){
        perror("DGRAM socket error"); exit(1); }
    else printf("opening datagram socket..... ok.\n");

    if(setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(yes))<0){
        perror("Reusing ADDR failed"); exit(1); }
    else printf("Reusing ADDR... ok.\n");
    bzero(&addr, 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) {
        perror("Bind error"); exit(1); }
    else printf("Binding datagram socket.... ok.\n");
```

Teacher's Signature

Date

Expt. No.

Page No.

```
mreq.imr_multiaddr.s_addr = inet_addr(HELLO_GROUP);
mreq.imr_interface.s_addr = htonl(INADDR_ANY);
if (setsockopt(fd, IPPROTO_IP, IP_ADD_MEMBERSHIP, (char*)&mreq,
      sizeof(mreq)) < 0) {
   perror("Adding multicast error"); exit(1); }
else printf("Adding multicast group.... ok.\n");
ml = sizeof(msgbuf);
while(1) {
   if ((nbytes = recvfrom(fd, msgbuf, sizeof(msgbuf), 0, NULL, NULL))<0){
      perror("Reading datagram message error"); exit(0); }

   printf("The message from multicast server is : %s \n", msgbuf);
   }
}
```

Sender

```
→ #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> #include<unistd.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
int main(int argc, char *argv[]) {
struct sockaddr_in addr;
  int fd, cnt;
  char* message;
  struct ip_mreq mreq;
  if(argc==1) message="PYCE-CSE";
   else message = argv[1];
```

Teacher's Signature

Date

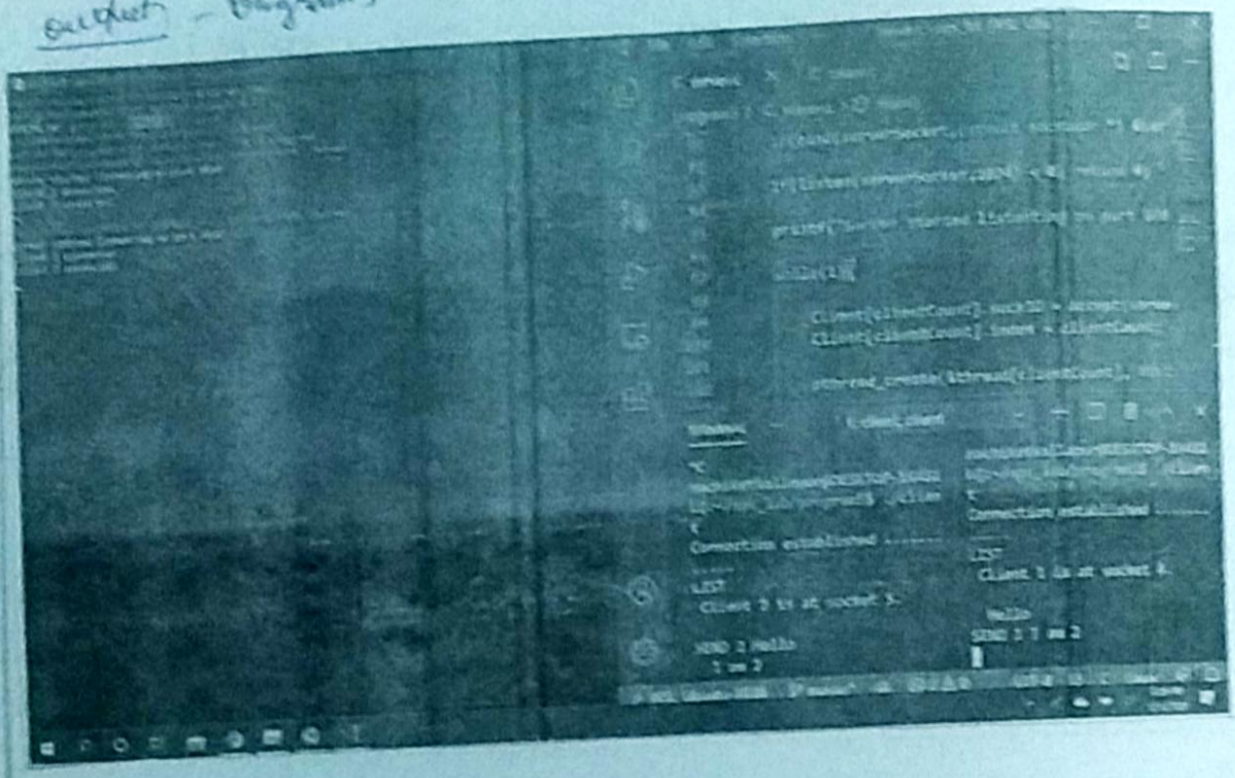
Expt. No.

Page No.

```
if((fd = socket (AF_INET, SOCK_DGRAM, 0)) <0) {
    perror("opening datagram socket error"); exit(1); }
else
  printf(" opening the datagram socket... ok. \n");
memset (&addr, 0, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr(HELLO_GROUP);
addr.sin_port = htons(HELLO_PORT);

while (1) {
  if ( sendto(fd, message, strlen(message), 0, (struct sockaddr*)&addr,
      sizeof(addr))<0){
      perror("sending datagram message error"); exit(1); }
  else printf(" Sending datagram message ... ok\n");
  sleep(2);
  }
return 0;
}
```

Teacher's Signature

Output - Program 5



Date

Expt. No. Page No.

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

→ Server

```
#include<sys/types.h> #include<sys/socket.h> #include<netinet/in.h>
#include<sys/stat.h> #include<unistd.h> #include<stdlib.h>
#include<stdio.h> #include<fcntl.h> #include<arpa/inet.h>
#include<string.h>
#define max 255
void str_echo(int connfd){
    int n;
    char *buffer = malloc(max);
    while ((n = recv(connfd, buffer, max, 0))>0) {
        fputs("From client: ", stdout);
        fputs("\n", stdout);
        if (fgets(buffer, max, stdin) != NULL) send(connfd, buffer, strlen(buffer), 0);
        bzero(buffer, max);
    }}
int main(){
    int cont, listenfd, connfd, addrlen, addrlen2, fd, pid, addrlen3;
    struct sockaddr_in address, cli_address;
    if ((listenfd = socket(AF_INET, SOCK_STREAM, 0))>0)
            printf("Socket was created");
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(1600);
    printf("Address before bind %s...\n", inet_ntoa(address.sin_addr));
    if (bind(listenfd, (struct sockaddr*) &address, sizeof(address))==0)
        printf("Binding socket\n");
    ~~printf~~ else {perror("Error in Bind"); exit(0);}
```

Teacher's Signature

Date

Expt. No.

Page No.

```
printf("The address after bind %s...\n", inet_ntoa(address.sin_addr));
listen(listenfd, 3);
printf("Server is listening");
getsockname(listenfd, (struct sockaddr*) &address, &addrlen);
printf("Server's local address %s... and port %d\n", inet_ntoa(address.sin_addr),
ntohs(address.sin_port));
for(;;){
  addrlen = sizeof(struct sockaddr_in);
  connfd = accept(listenfd, (struct sockaddr*) &cli_address, &addrlen);
  addrlen2 = sizeof(struct sockaddr_in);
  int i = getpeername(connfd, (struct sockaddr*) &cli_address, &addrlen);
  printf("client %s connected on port %d \n", inet_ntoa(cli_address.sin_addr),
     ntohs(cli_address.sin_port));
  if((pid = fork()) == 0){
      printf("Inside child"); printf("\n");
      close(listenfd);
      strecho(connfd);
      exit(0);
    }
  close(connfd);
}
return 0;
}
```

Client

```
#include<sys/socket.h> #include<sys/types.h> #include<netinet/in.h>
#include<unistd.h> #include<stdlib.h> #include<stdio.h> #include<string.h>
#include<arpa/inet.h>
#define max 255
```

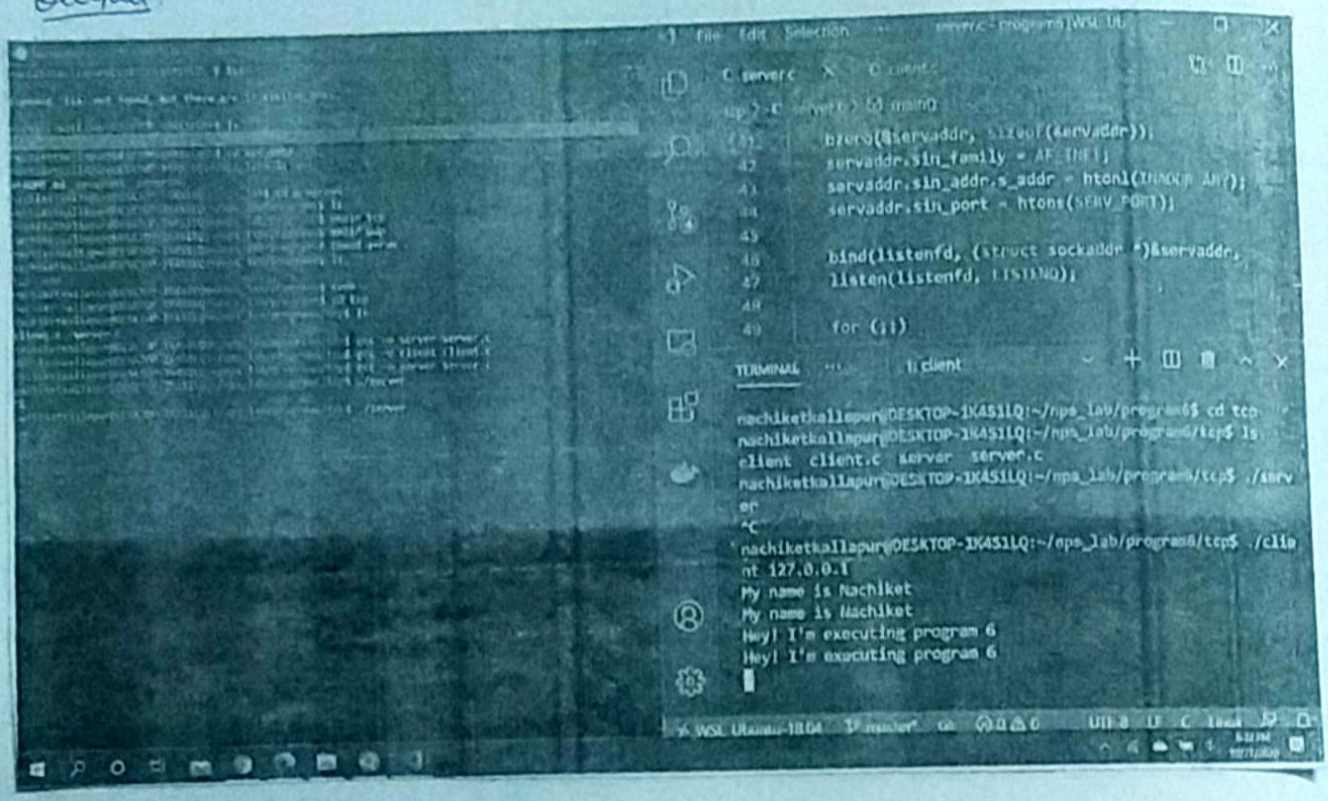
Teacher's Signature

Expt. No. _____ Page No. _____

```
void str_cli(FILE *ff, int sockfd){
  int cont; char *buffer = malloc(max); fputs("To Server: ", stdout);
  while (fgets (buffer, max, ff) != NULL) {
    send(sockfd, buffer, strlen (buffer), 0);
    if ((cont = recv(sockfd, buffer, max, 0)) > 0){
        fputs("From server: ", stdout); fputs( buffer, stdout);
    if if(strcmp (buffer, "exit", 4)) == 0) {
            printf ("Client exit...\n"); break; }
        bzero(buffer, 10240); }
    fputs ("\n\t To Server: ", stdout);
  }
  printf ("\nEOF\n"); }
int main (int argc, char *argv[]){
  int create_socket; struct sockaddr_in address;
  if((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
      printf (" Socket created successfully");
  address.sin_family = AF_INET; address.sin_port = htons(8001);
  inet_pton(AF_INET, argv[1], &address.sin_addr);
  if (connect (create_socket, (struct sockaddr*) &address,
        sizeof (address)) == 0)
    printf (" Connection accepted with the server %s\n", argv[1]);
  else { printf ("Error in connect"); }

  str_cli (stdin, create_socket);
  return close(create_socket);
}
```

Teacher's Signature _____

Output - TCP concurrent



Expt. No. _____ Page No. _____

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

→ i) TCP Concurrent:

Client:

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

void str_cli(FILE *fp, int sockfd){
    int buffersize = 1024;
    char *buffer = malloc(buffersize);
    while(fgets(buffer, buffsize, fp) != NULL) {
        if ((strncmp(buffer, "exit", 4)) == 0)
            printf("Client Exit \n");
        send(sockfd, buffer, sizeof(buffer), 0);
        if (recv(sockfd, buffer, buffsize, 0) > 0)
            fputs(buffer, stdout);
    }
    printf("\n EOF \n");
    free(buffer); }

int main(int argc, char *argv[]){
    int create_socket;
    struct sockaddr_in address;
    if ((create_socket = socket(AF_INET, SOCK_STREAM, 0)) > 0)
        printf("Socket was created \n");
    address.sin_family = AF_INET;
    address.sin_port = htons(15000);
    inet_pton(AF_INET, argv[1], &address.sin_addr);
```

Teacher's Signature _____

Date

Expt. No.

Page No.

```
if (connect(create_socket, (struct sockaddr *) &address, sizeof(address))==0)
    printf("Connection accepted with the server%s...\n", argv[0]);
else{ printf("Error in connect"); exit(0);

str_cli(stdin, create_socket);
return close(create_socket);
}
```

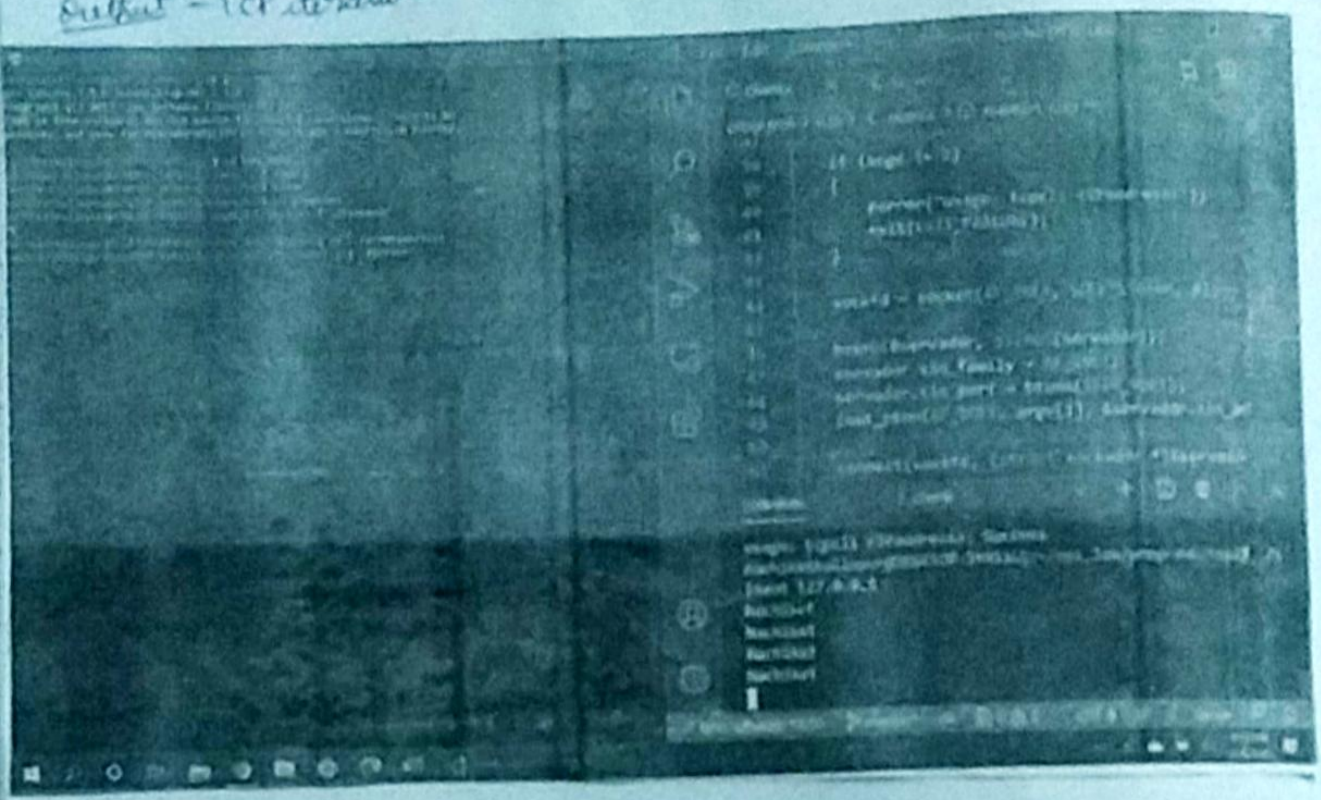
server

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

void str_echo(int connfd){
    int n;
    int buffsize = 1024;
    char *buffer = malloc(buffsize);
again:
    while((n= recv(connfd, buffer, buffsize, 0))>0)
        send(connfd, buffer, n, 0);
    if (n<0) goto again;

    free(buffer);
}

int main(){
  int listenfd, connfd, addrlen, pid, addrlen3;
  struct sockaddr_in address, cli_address;
  if ((listenfd = socket(AF_INET, SOCK_STREAM, 0))>0)
        printf("Socket was created\n");
  address.sin_family= AF_INET; address.sin_addr.s_addr=INADDR_ANY;
```

Teacher's Signature

```
address.sin_port = htons(15001);
printf("Address before bind %s...\n", inet_ntoa(address.sin_addr));
if(bind(listenfd, (struct sockaddr*) &address, sizeof(address)) == 0)
    printf("Binding Socket\n");
printf("Address after bind %s...\n", inet_ntoa(address.sin_addr));
listen(listenfd, 3);
printf("Server is listening");
getsockname(listenfd, (struct sockaddr*) &address, &addrlen);
printf("The server's local address %s and port %d\n",
    inet_ntoa(address.sin_addr), ntohs(address.sin_port));
for(;;){
    addrlen = sizeof(struct sockaddr_in);
    connfd = accept(listenfd, (struct sockaddr*) &cli_address, &addrlen);
    int p = getpeername(connfd, (struct sockaddr*) &cli_address, &addrlen);
    if(connfd > 0){
        printf("Client %s connected on port %d\n", inet_ntoa(cli_address.sin_addr),
            ntohs(cli_address.sin_port));
    if(pid = fork()) == 0){
        printf("Inside child\n"); close(listenfd); strcpy(connfd);
        exit(0); }

    close(connfd);
}
return 0;
}
```

Output - TCP iterative



Date

Expt. No. Page No.

TCP-Iterative

Server

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

void str_echo(int connfd){
    int n;
    int buffsize=1024;
    char * buffer= malloc(buffsize);
again:
    while((n=recv(connfd, buffer, buffsize, 0))>0)
        send(connfd, buffer, n, 0);
    if(n<0)
        goto again;
}

int main(){
    int cont, listenfd, connfd, addrlen, addrlen2, fd, pid, addrlen3;
    struct sockaddr_in address, cli_address;
    if((listenfd = socket(AF_INET, SOCK_STREAM, 0))>0)
        printf("Socket was created \n");
    address.sin_family = AF_INET;
    address.sin_addr.s_addr= INADDR_ANY;
    address.sin_port= htons(15001);
    printf("Address before bind is %s...\n", inet_ntoa(address.sin_addr));
    if(bind(listenfd, (struct sockaddr*) &address, sizeof(address))==0)
        printf("Binding Socket\n");
    printf("Address after bind is %s...\n", inet_ntoa(address.sin_addr));
```

Teacher's Signature

Date

Expt. No.

Page No.

```
listen(listenfd, 5);
printf("Server is listening\n");

for(;;){
  addrlen= sizeof (struct sockaddr_in);
  connfd = accept (listenfd, (struct sockaddr*),&cli_address, &addrlen);
  printf("Client %s is connected on port %d\n",
    inet_ntoa(cli_address.sin_addr) htons(cli_address.sin_port));

  strecho(connfd);
  close (connfd);
}
  return 0;
}
```

Client

```
#include <stdio.h> #include <stdlib.h> #include <sys/types.h>
#include <sys/socket.h> #include <netinet/in.h> #include <fcntl.h>
#include <unistd.h> #include <sys/stat.h> #include <arpa/inet.h>
#include <string.h>
void str_cli(FILE * fp, int sockfd){
  int buffsize=1024,cont;
  char * buffer = malloc(buffsize);
  while (fgets(buffer, buffsize, fp) != NULL){
      if( (strncmp (buffer, "exit",4))==0){
          printf("Client Exit ...\n"); break; }
    send (sockfd, buffer, sizeof (buffer),0);
    if( (cont = recv (sockfd, buffer, buffsize,0)) >0)
        fputs (buffer, stdout);
  }
```

Teacher's Signature

Date

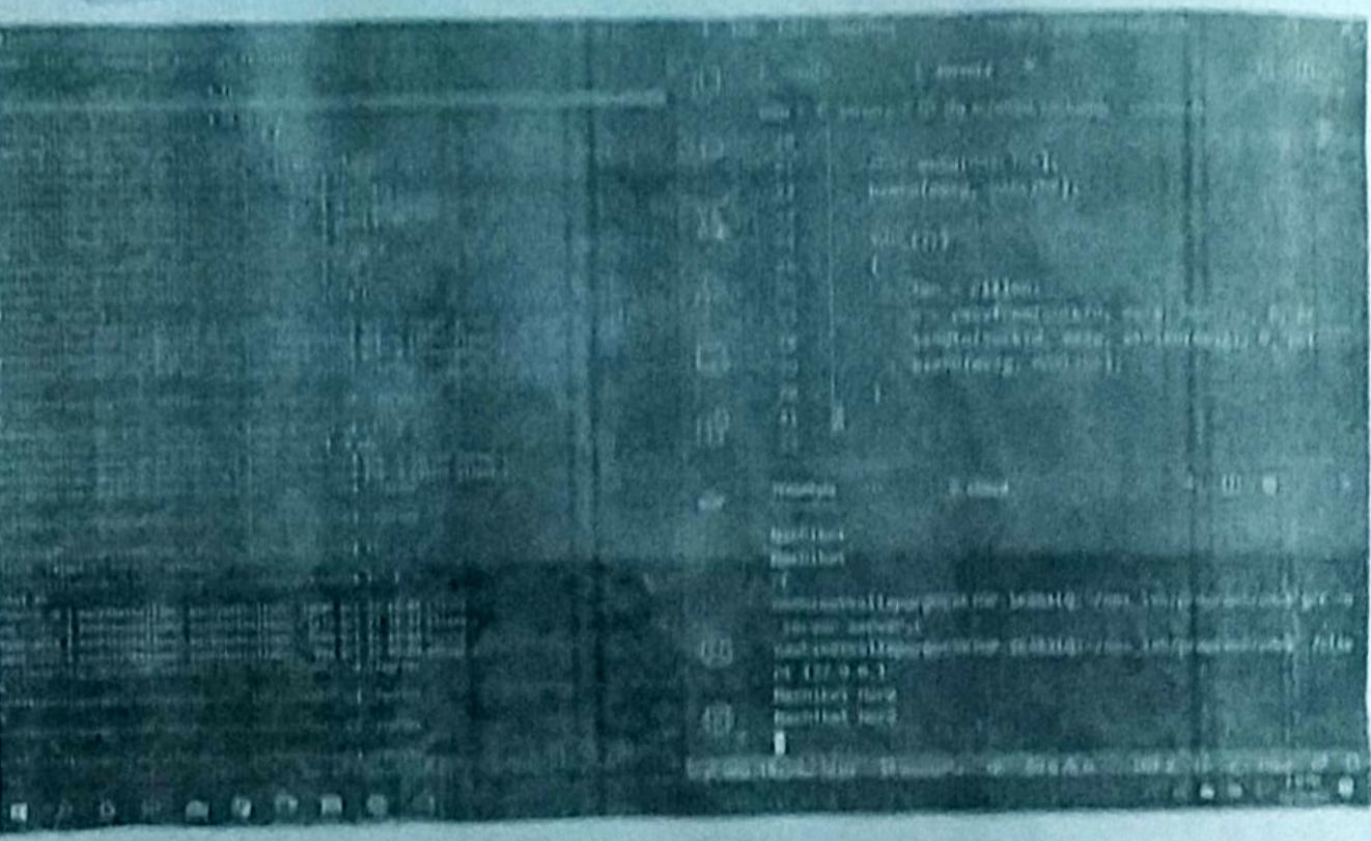
Expt. No.

Page No.

```
printf("EOF\n"); }

int main(int argc, char *argv[]){
  int create_socket;
  struct sockaddr_in address;
  if ((create_socket = socket(AF_INET, SOCK_STREAM, 0))>0)
     printf("Socket created\n");

  address.sin_family = AF_INET;
  address.sin_port = htons(15001);
  inet_pton(AF_INET, argv[1], &address.sin_addr);

  if(connect(create_socket, (struct sockaddr*) &address,
        sizeof(address)) == 0)
     printf("The client is connected to the server %s... \n", argv[1]);
  else {
    printf("Error in connect \n");
    exit(0);
  }
  str_cli(stdin, create_socket);
  return close(create_socket);
}
```

Teacher's Signature

output - UDP client/server



Expt. No. _____ Page No. _____

UDP-Iterative

Server

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

void str_echo(int sockfd, struct sockaddr* cli_address, int clilen){
    int n, buffsize = 1024; int addrlen;
    char* buffer = malloc(buffsize);
    for(;;){
        addrlen = clilen;
        n = recvfrom(sockfd, buffer, buffsize, 0, cli_address, &addrlen);
        sendto(sockfd, buffer, n, 0, cli_address, addrlen);
    }

int main(){
    int sockfd;
    struct sockaddr_in serv_address, cli_address;
    if (sockfd = socket(AF_INET, SOCK_DGRAM, 0)) >0)
        printf("Socket was created\n");
    serv_address.sin_family = AF_INET;
    serv_address.sin_addr.s_addr = INADDR_ANY;
    serv_address.sin_port = htons(16001);
    printf("The address before bind is %s...\n",
        inet_ntoa(serv_address.sin_addr));
    if(bind(sockfd, (struct sockaddr*)&serv_address,
        sizeof(serv_address))==0)
        printf("Binding Socket\n");
```

Teacher's Signature _____

Date

Expt. No. Page No.

```
str_echo(sockfd, (struct sockaddr*) &cli_address, sizeof(cli_address));
return 0;
}
```

client

```
#include<sys/socket.h> #include<sys/types.h> #include<netinet/in.h>
#include<unistd.h> #include<stdlib.h> #include<stdio.h>
#include<arpa/inet.h> #include<string.h> #include<errno.h>
#define SERV_PORT 9002
#define MAXLINE 1024

void dg_cli(FILE *fp, int sockfd, const struct sockaddr* pservaddr,
    socklen_t servlen) {
  int n;
  char sendline[MAXLINE], recvline[MAXLINE];

  while(fgets(sendline, MAXLINE, fp) != NULL){
    sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);
    if(strncmp(sendline, "exit", 4) == 0){
        printf("Client Exit\n");
        break;
    }
    n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
    recvline[n] = '\0';
    fputs(recvline, stdout);
  }
}
```

Teacher's Signature

Date

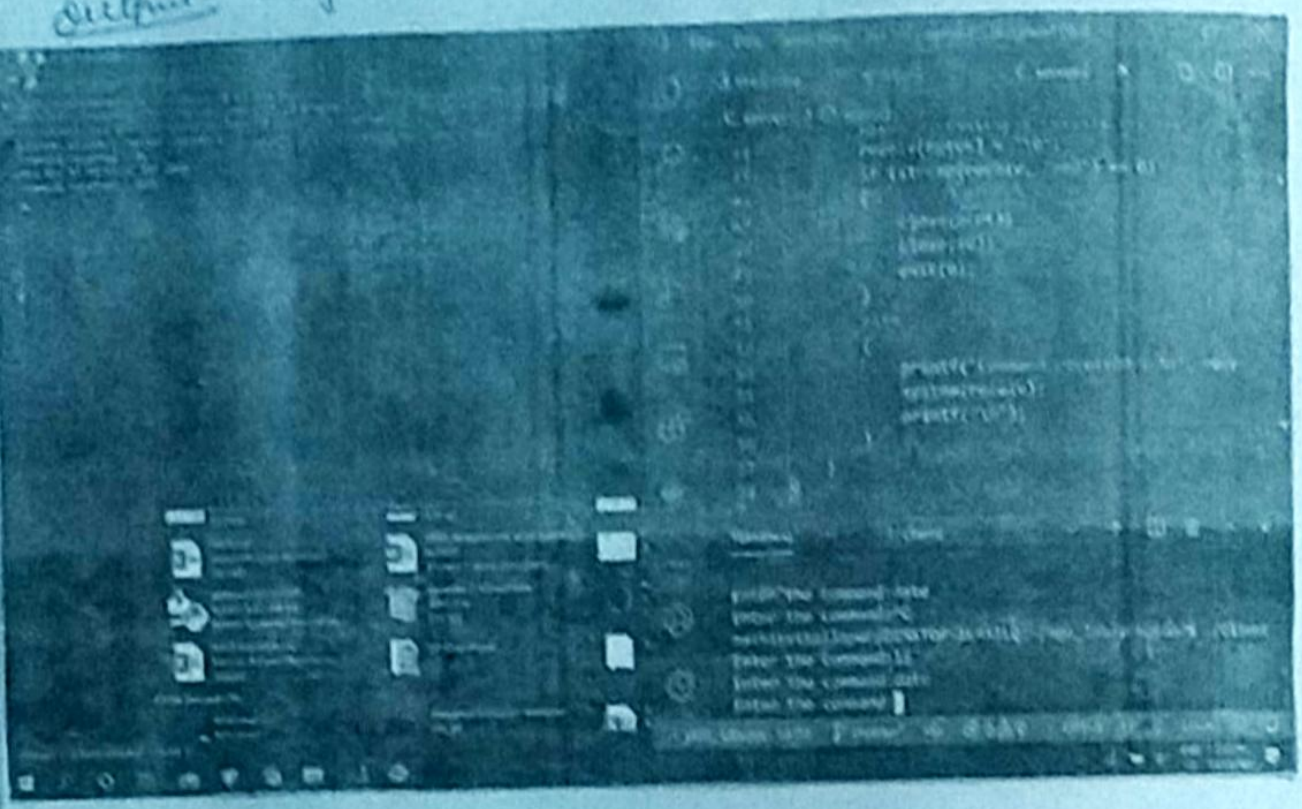
Expt. No.

Page No.

```
int main( int argc, char *argv[]){
  int sockfd;
  struct sockaddr_in serv_address;
  if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) > 0)
     printf("Socket was created \n");
  serv_address.sin_family = AF_INET;
  serv_address.sin_port = htons(16001);

  inet_pton(AF_INET, argv[1], &serv_address.sin_addr);

  dg_cli(stdin, sockfd, (struct sockaddr*) &serv_address,
     sizeof(serv_address));
  exit(0);
}
```

Teacher's Signature

Output — Bug ?



Date

Expt. No.

Page No.

a) Implementation of remote command execution using socket system calls.

→ Server

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

int main(){
    int sd, cnfd, len, bytes, port;
    char send[50], recv[50];
    struct sockaddr_in serv,cli;
    if ((sd = socket(AF_INET, SOCK_STREAM, 0))>0) {
        printf("Error in socket\n");
        exit(0);
    }
    else
        printf("Socket created successfully...\n");

    bzero(&serv, sizeof(serv));
    serv.sin_family = AF_INET;
    serv.sin_port = Htons(8080);
    serv.sin_addr.s_addr = Htonl(INADDR_ANY);
    if (bind(sd, (struct sockaddr*)&serv, sizeof(serv)) <0)
        printf("Error in bind\n");
        exit(0);
    }
    else printf("Binding Socket to IP: %s\n", inet_ntoa(serv.sin_addr));
    listen(sd,3);
```

Teacher's Signature

Date

Expt. No. Page No.

```
if ((acpt= accept(sd,(struct sockaddr*)NULL,NULL))<0) {
    printf("Error in accept\n");
    exit(0);
}
else printf("Client connected...\n");
while(1){
    bytes= recv(acpt, receive, 50,0);
    receive[bytes]='\0';
    if (strcmp(receive,"end")==0) {
        close(acpt);
        close(sd);
        exit(0);
    }
    else{
        printf("Command received : %s", receive);
        system(receive);
        printf("\n");
    }
}
}
```

Client

```
> #include <stdio.h> #include<stdlib.h> #include<string.h>
#include <unistd.h> #include<netinet/in.h> #include<arpa/inet.h>
#include<sys/types.h> #include<sys/socket.h> #include<errno.h>
int main(){
    int sd, acpt, len, bytes, port;
    char send[50], receive[50];
    struct sockaddr_in serv_cli;
```

Teacher's Signature

Date

Expt. No.

Page No.

```
if((sd=socket(AF_INET, SOCK_STREAM, 0))<0) {
    printf("Error in socket\n");
    exit(0);
}
else printf("Socket created successfully...\n");
bzero(&serv, sizeof(serv));
serv.sin_family = AF_INET;
serv.sin_port= htons(8080);
serv.sin_addr.s_addr= htonl(INADDR_ANY);
if(connect(sd, (struct sockaddr*)&serv, sizeof(serv)) <0) {
    printf("Error in connection \n");
    exit(0);
}
else printf("Connection established with the server...\n");
while(1){
printf("Enter the command: "); gets(send1);
if (strcmp(send1,"end")!=0) send(sd, send1, 50,0);
else {
    send(sd, send1, 50, 0);
    close(sd);
    break;
}
}
}
```

Teacher's Signature

output - RSA

```
rsa.cpp
#include <iostream>
#include <cstdlib.h>
#include <math.h>
#include <string.h>
using namespace std;
long int gcd(long int a, long int b)
{
if(a == 0)
return b;
if(b == 0)
return a;
return gcd(b, a%b);
```

```
nachikettkallapur@DESKTOP-1K45I1Q:~/cpp_lab/program8$ g++ -o rsa rsa.cpp
nachikettkallapur@DESKTOP-1K45I1Q:~/cpp_lab/program8$ ./rsa
Enter the text to be encrypted: Nachiket
Two prime numbers (p and q) are: 13 and 23
n(p * q) = 13 * 23 = 299
(p - 1) * (q - 1) = 264
Public key (n, e): (299, 101)
Private key (n, d): (299, 223)
Encrypted message: 52111831324815975116
Decrypted message: Nachiket
nachikettkallapur@DESKTOP-1K45I1Q:~/cpp_lab/program8$
```

Date

Expt. No. Page No.

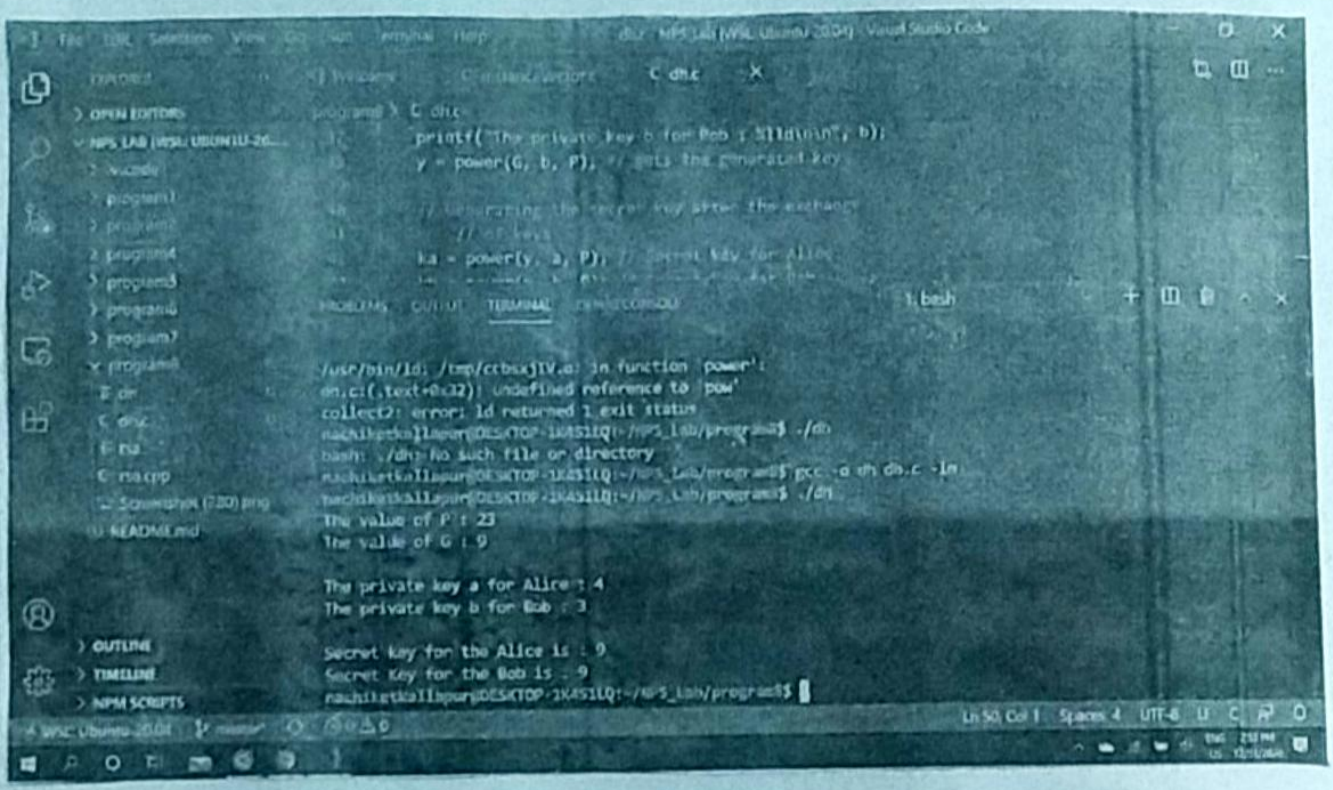
8) write a program to encrypt and decrypt the data using RSA and exchange the key securely using Diffie-Hellman Key exchange protocol

→ RSA

```
#include <iostream> #include <stdlib.h> #include <math.h>
#include <string.h>
using namespace std;

long ~~long~~ int gcd (long int a, long int b) {
    if (a==0) return b;
    if (b==0) return a;
    return gcd (b, a%b);
  }
long int isprime(long int a) {
    int i;
    for (i=2; i<a; i++) {
        if ((a%i)==0)
            return 0;
    }
    return 1;
}
long int encrypt (char ch, long int n, long int e) {
    int i;
    long int temp = ch;
    for (i=1; i<e; i++)
        temp = (temp * ch)%n;
    return temp;
}
```

Teacher's Signature

output- diffiehellman



Date

Expt. No. Page No.

```
char decrypt(long int ch, long int n, long int d){
    int i;
    long int temp=ch;
    for(i=1; i<d; i++)
        ch=(temp*ch)%n;
    return ch;
}
int main(){
    long int i, len;
    long int p, q, n, phi, e, d, cipher[50];
    char text[50];
    cout<<"Enter ~~time~~ the text to be encrypted: ";
    cin.getline(text, sizeof(text));
    len=strlen(text);
    do{
        p=rand()%50;
    }while(!isprime(p));
    do{
        q=rand()%50;
    }while(!isprime(q));
    n=p*q;
    phi=(p-1)*(q-1);
    do{
        e=rand%phi;
    }while(gcd(phi,e)!=1);
    do{
        d=rand()%phi;
    }~~while(gcd(phi,e)!=1)~~; while((d*e%phi)!=1);
```

Teacher's Signature

Date

Expt. No.

Page No.

```
cout<<"Two prime number (p and q) are : "<<p<<"and"<<q<<endl;
cout<<" n(p*q)= "<<p<<"*"<<q<<"*"<<p*q<<endl;
cout<<"(p-1)*(q-1)"= "<<phi<<endl;
cout<<" Public Key (n,e) : ("<<n<<","<<e<<")\n";
cout<<"Private Key(n,d) : ("<<n<<","<<d<<")\n";
for(i=0; i<len; i++)
    cipher[i]=encrypt(text[i],n,e);
cout<<" Encrypted message: ";
for(i=0; i<len; i++)
    cout<<cipher[i];
for(i=0; i<len; i++)
    text[i]= decrypt(cipher[i],n,d);
cout<<endl;
return 0;
}
```

Diffie Hellman

```
#include<stdio.h>
#include<math.h>
long long int power(long long int a, long long int b, long long int p){
    if(b==1) return a;
    else return ((long long int)pow(a,b))%p;
}

int main(){
    long long int p, g, x, a, y, b, ka, kb;
    p=23;
    printf("The value of p: %ddd\n",p);
```

Teacher's Signature

Date

Expt. No.

Page No.

```
G=9;
printf("The value of G: %lld\n", G);
a=4;
printf(" The private key a for Alice : %lld\n", a);
x = power(G, a, P);

b=3;
printf("The private key b for Bob : %lld\n\n", b);
y = power(G, b, P);

ka= power(y, a, P);
kb = power(x, b, P);

printf("Secret key for the Alice is : %lld\n", ka);
printf("Secret key for the Bob is : %lld\n", kb);

return 0;
}
```

Teacher's Signature

Part B-1a

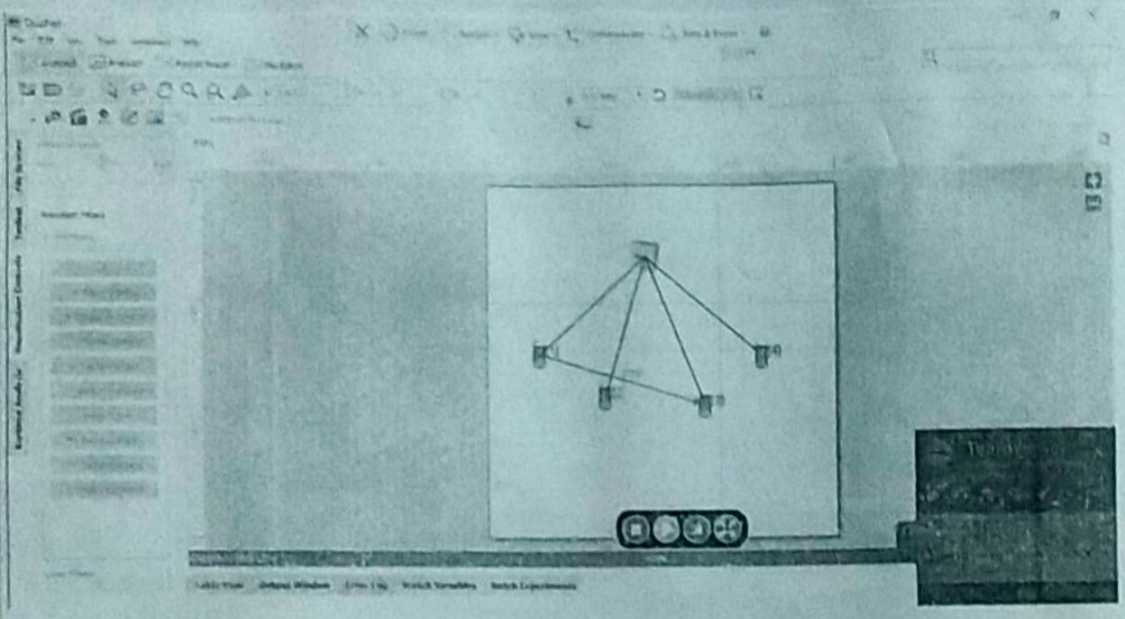


Fig 1

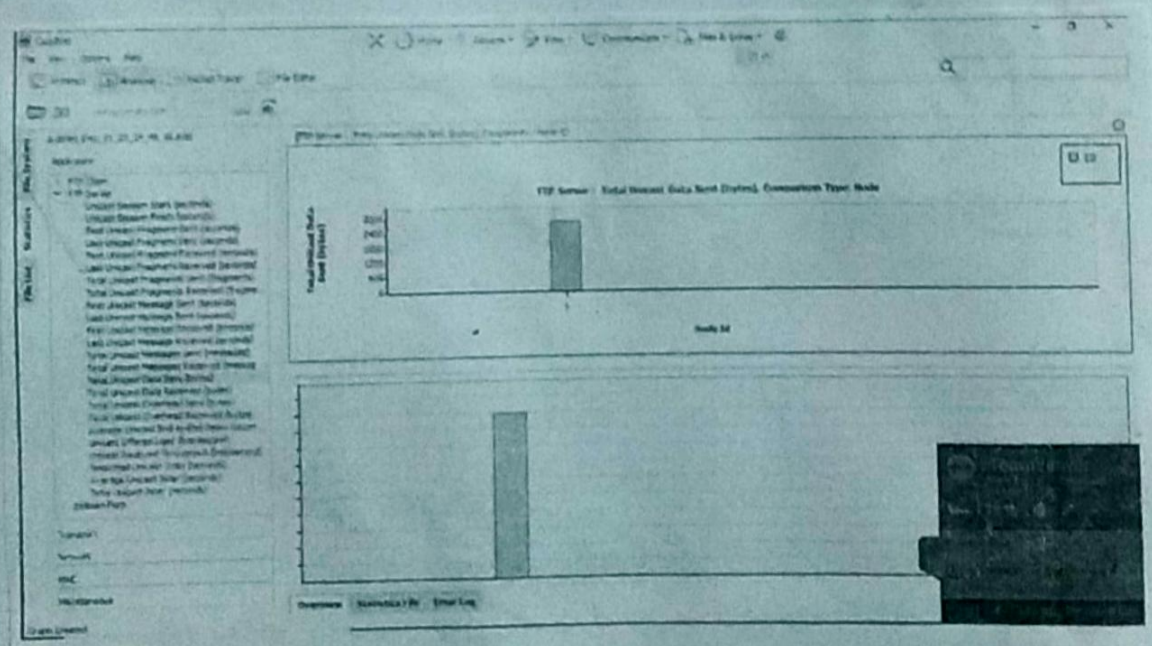


Fig 2

Date

Expt. No.

Page No.

Part-B

1) Setup an IEEE 802.3 network with
a) Hub b) Switch c) Hierarchy of switch. Apply the FTP, Telnet applications between nodes. Vary the number of nodes. Vary the bandwidth, queue size and observe the packet drop probability.

Fig 1. Network architecture with 4 devices and a switch
Device 1 and Device 3 exchange packets using FTP.

Fig 2. Total Unicast data sent (in bytes) by FTP Server.

Teacher's Signature

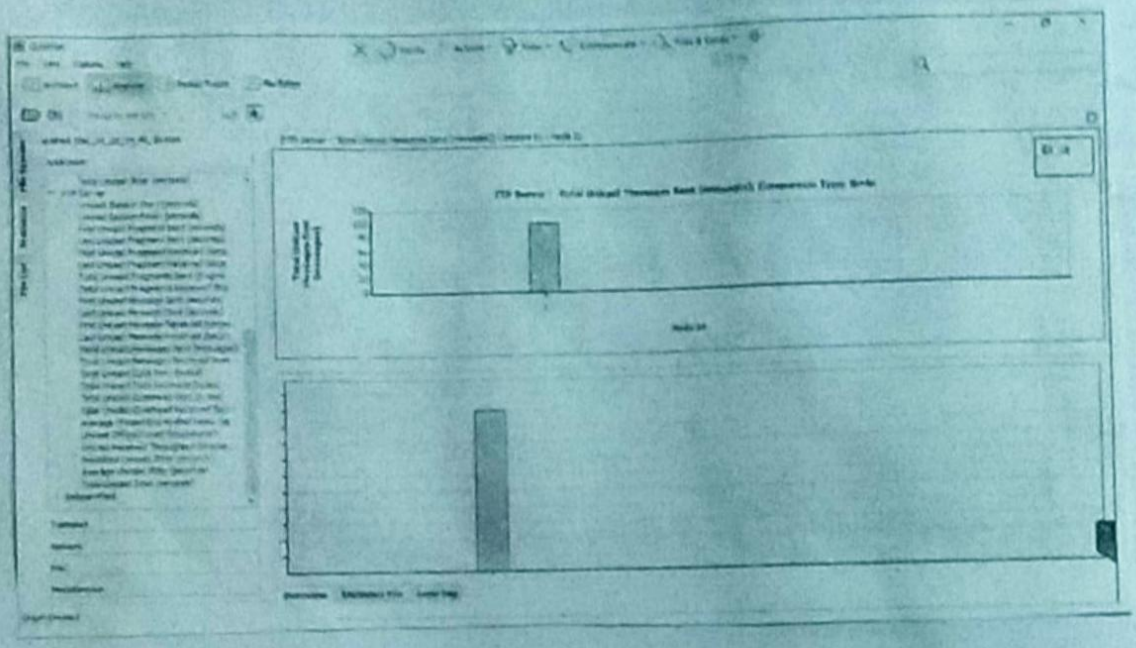


Fig 3

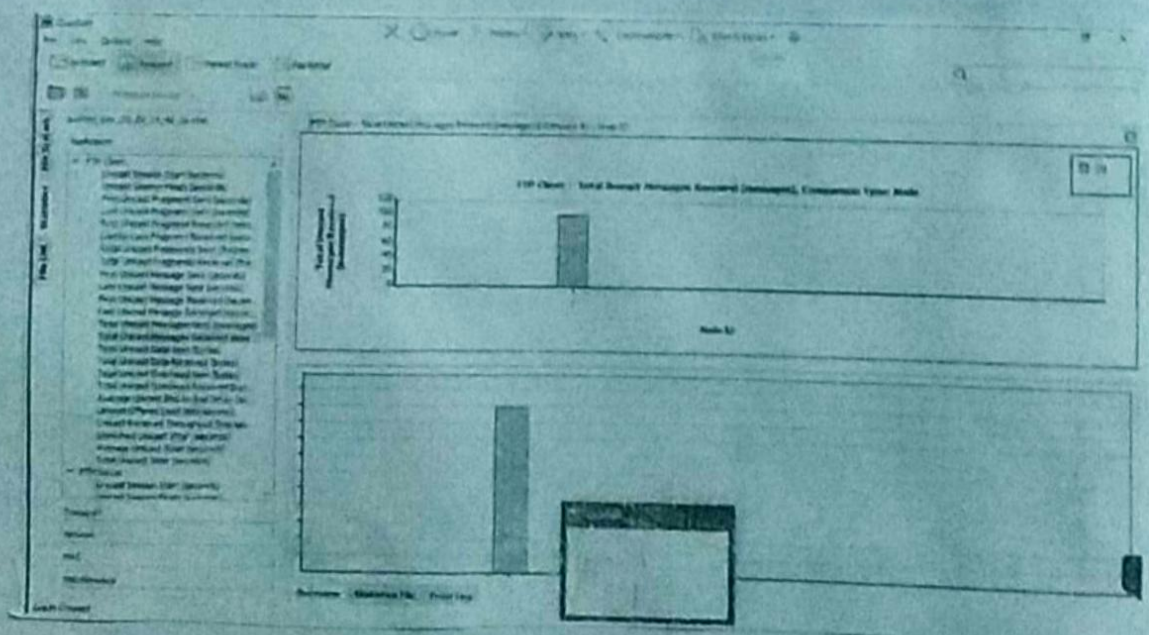


Fig 4

Date

Expt. No.

Page No.

Fig3: Total Unicast messages sent by FTP server

Fig4: Total Unicast messages received by TCP client

Teacher's Signature

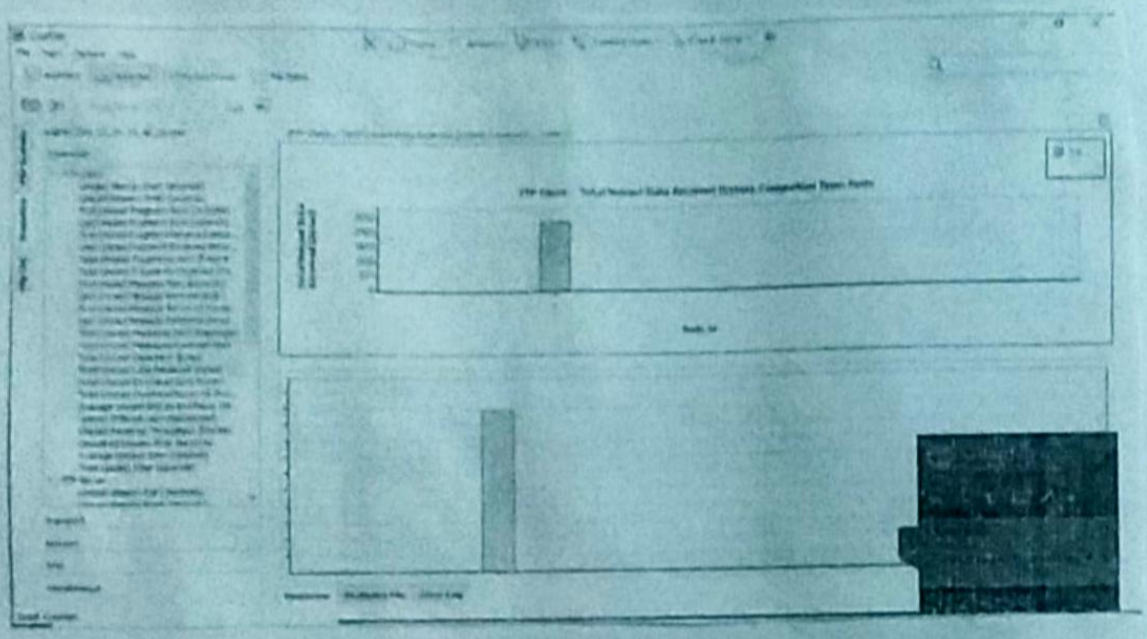


Fig 5

Date

Expt. No.

Page No.

Fig 5: Total Unicast data Received (in bytes) by FTP client.

Teacher's Signature

Part B - 1b

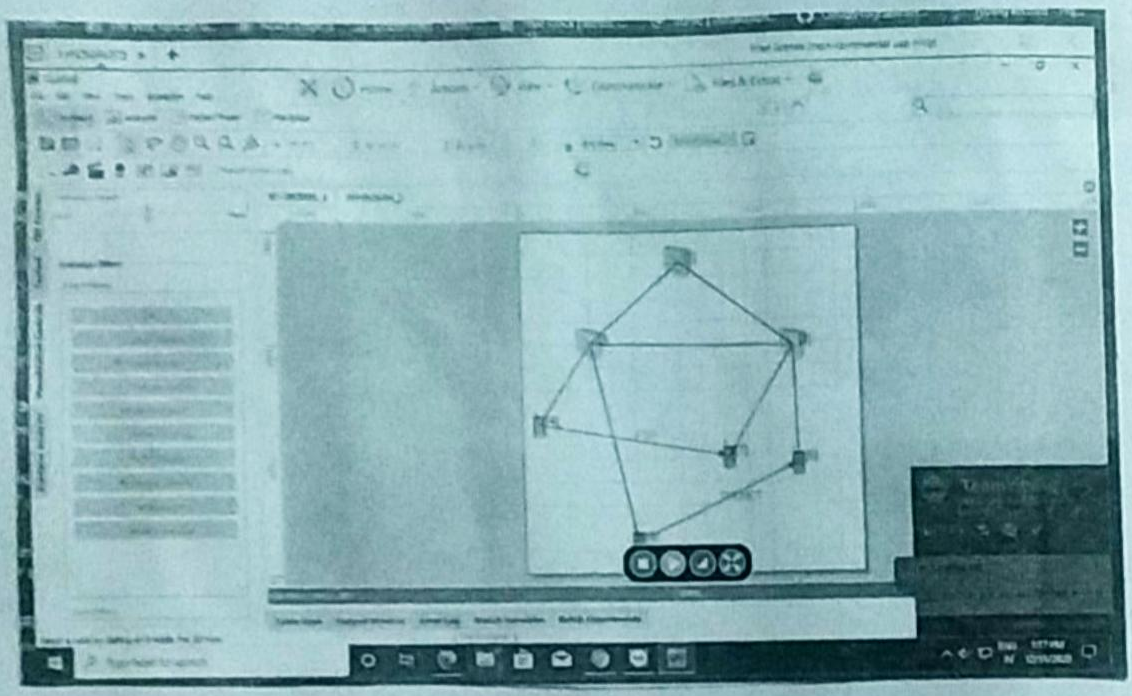


Fig1

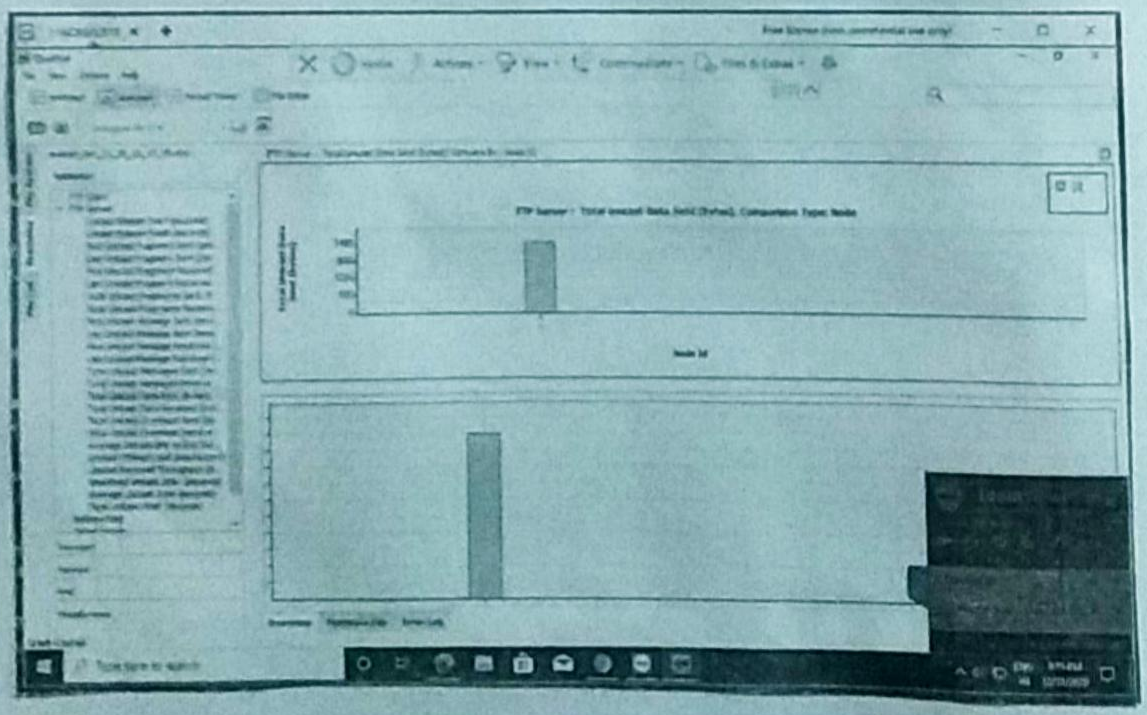


Fig2

Date

Expt. No.

Page No.

16)

~~Fig1: Network hierarchy with~~

Fig1: Network architecture with two hierarchy of switch and 4 devices. There is FTP communication between devices 4 and 7 and TELNET communication between devices 5 and 6.

Fig 2: Total unicast data (in bytes) sent by FTP server.

Teacher's Signature

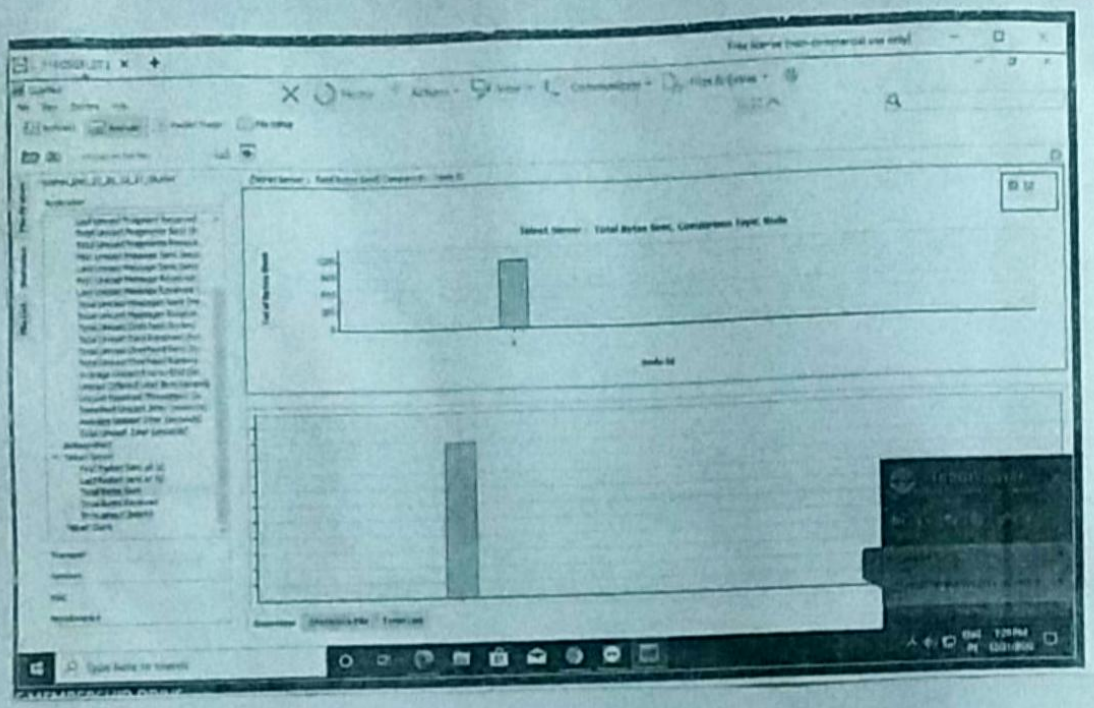


Fig3

Date

Expt. No.

Page No.

Fig3: Total bytes sent by Telnet Server

Teacher's Signature

PartB-2

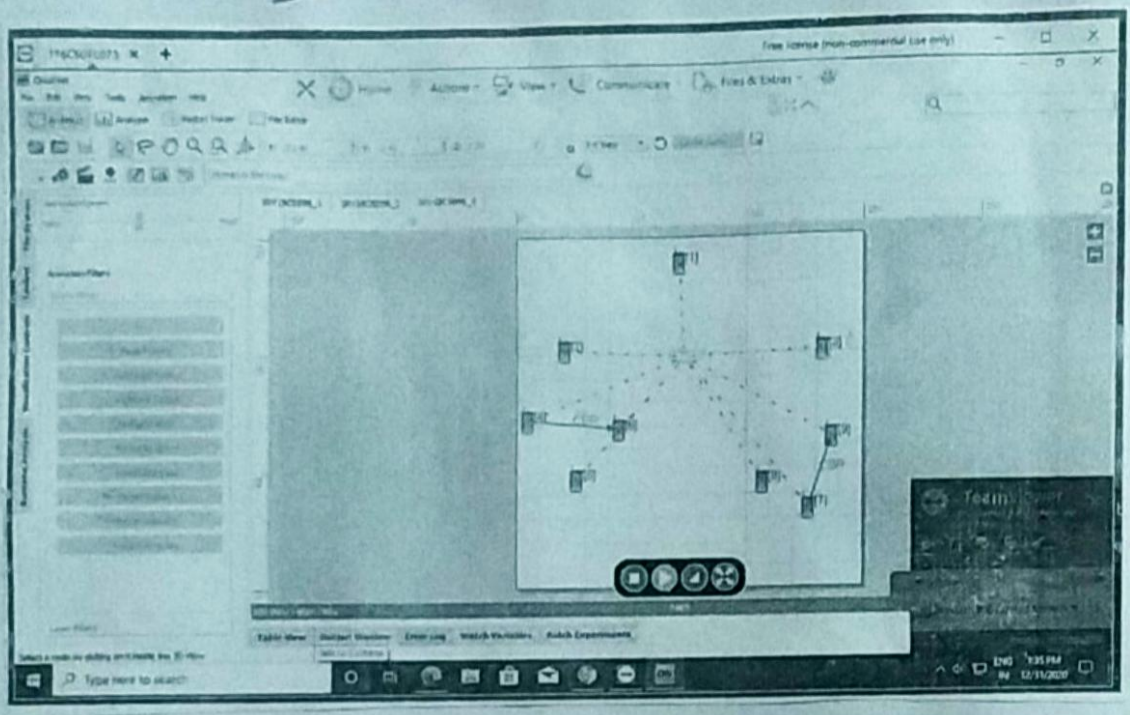


Fig1

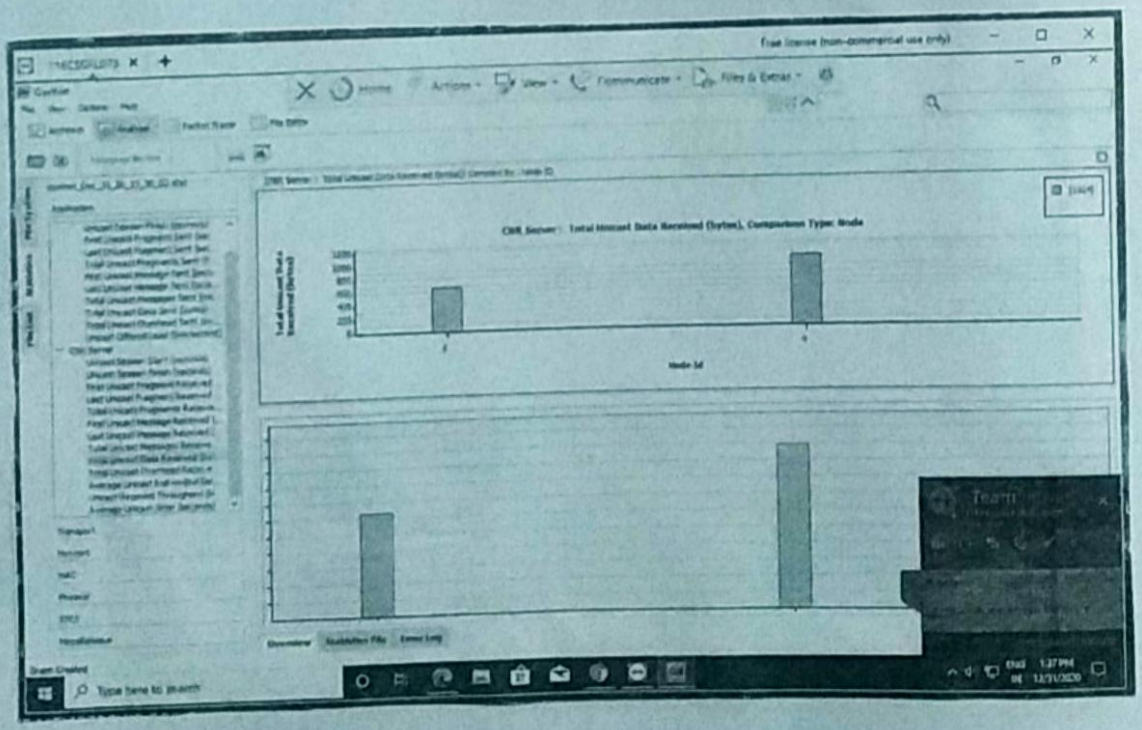


Fig2

Expt. No. _____ Page No. _____

2) Setup a wireless sensor networks with atleast two device coordinators and nodes. Provide Constant Bit Rate (CBR), Variable Bit Rate (VBR) application between several nodes. Increase the number of coordinators and nodes in the same area and observe the performance at physical and MAC layers.

Fig1 : network architecture of wireless LAN. Here device 1 is the coordinator, devices 2 and 5 are PAN coordinator

Fig2 : Total Unicast data Received (Bytes) by CBR server.

Teacher's Signature _____

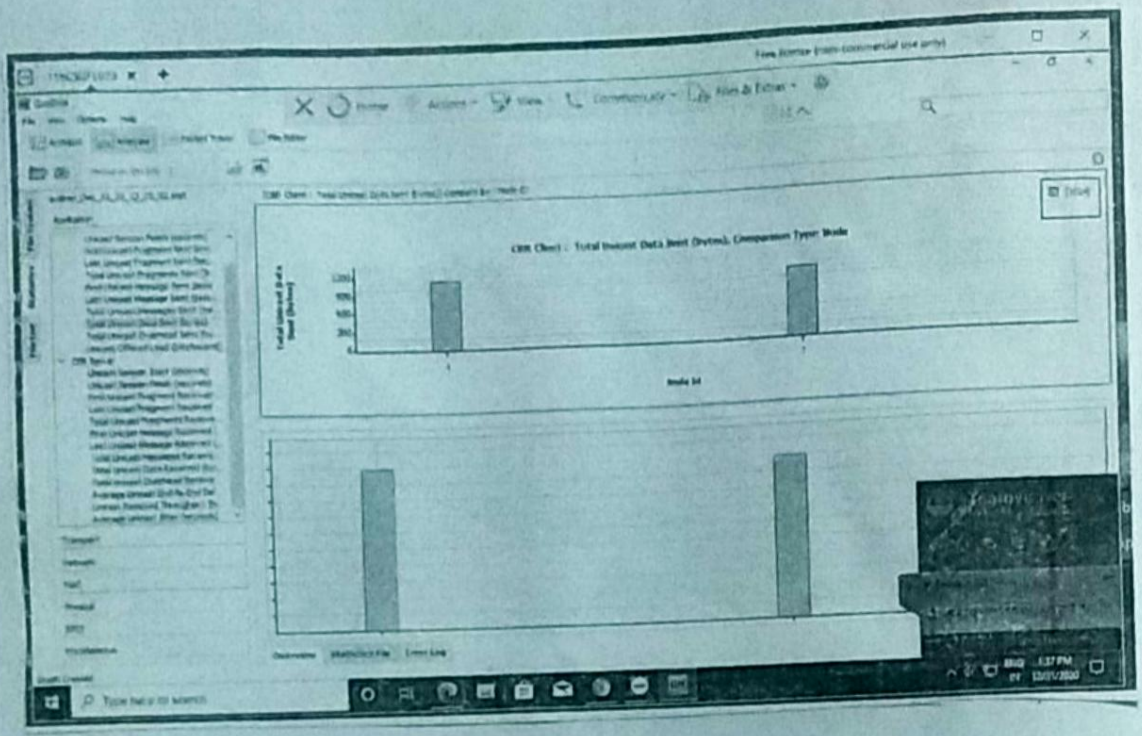

CBR Chart : Total Unicast Data Sent (bytes), Comparison Type: Node
Node Id

Fig3:

Date

Expt. No.

Page No.

Fig 3: Total Unicast data sent (bytes) by CBR client

Teacher's Signature

Part B -3

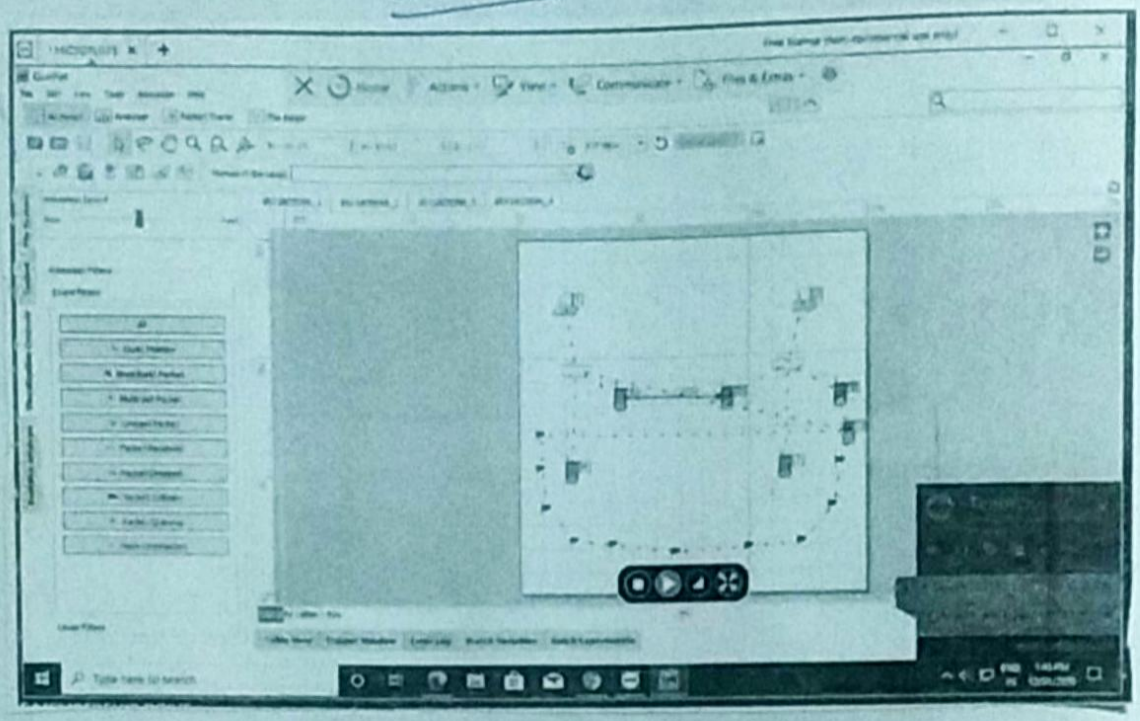


Fig1

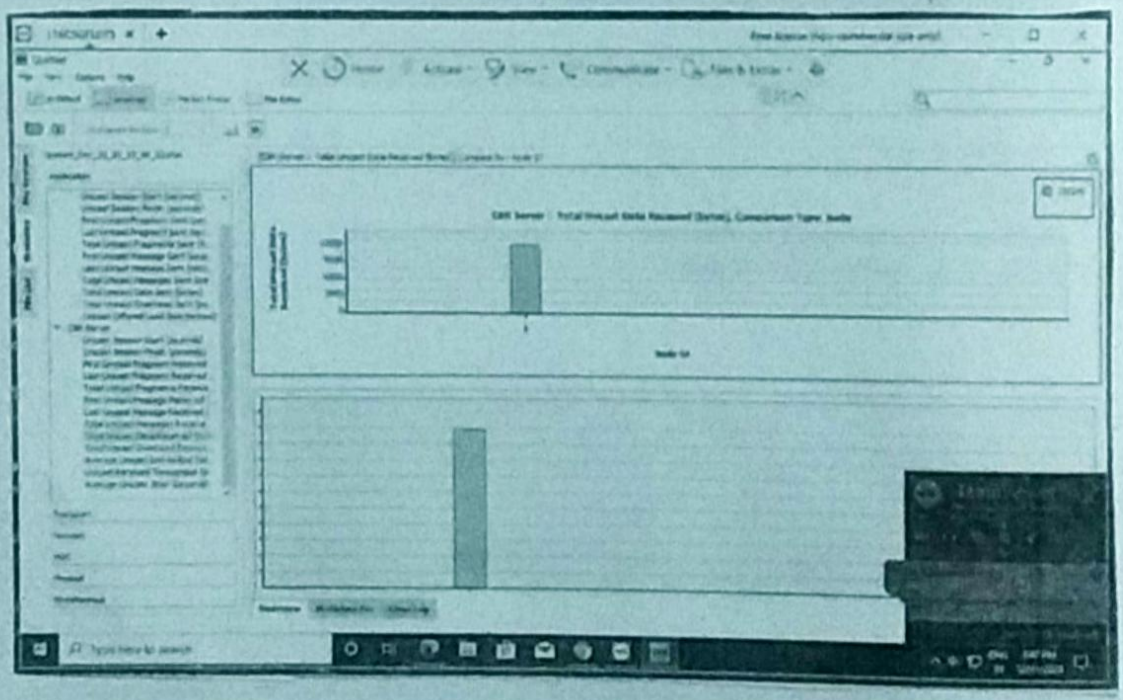


Fig2

Date

Expt. No.

Page No.

3) setup an IEEE 802.11 network with at least two access points. Apply the CBR, VBR applications between devices belonging to same access points and different access points. Provide roaming of any device, vary the numbers of access points and devices. Find out the delay in MAC layer, packet drop probability

Fig1: Network Architecture of IEEE 802.11 wireless LAN. It has two access points

Fig2: Total unicast data received (in bytes) by CBR server.

Teacher's Signature

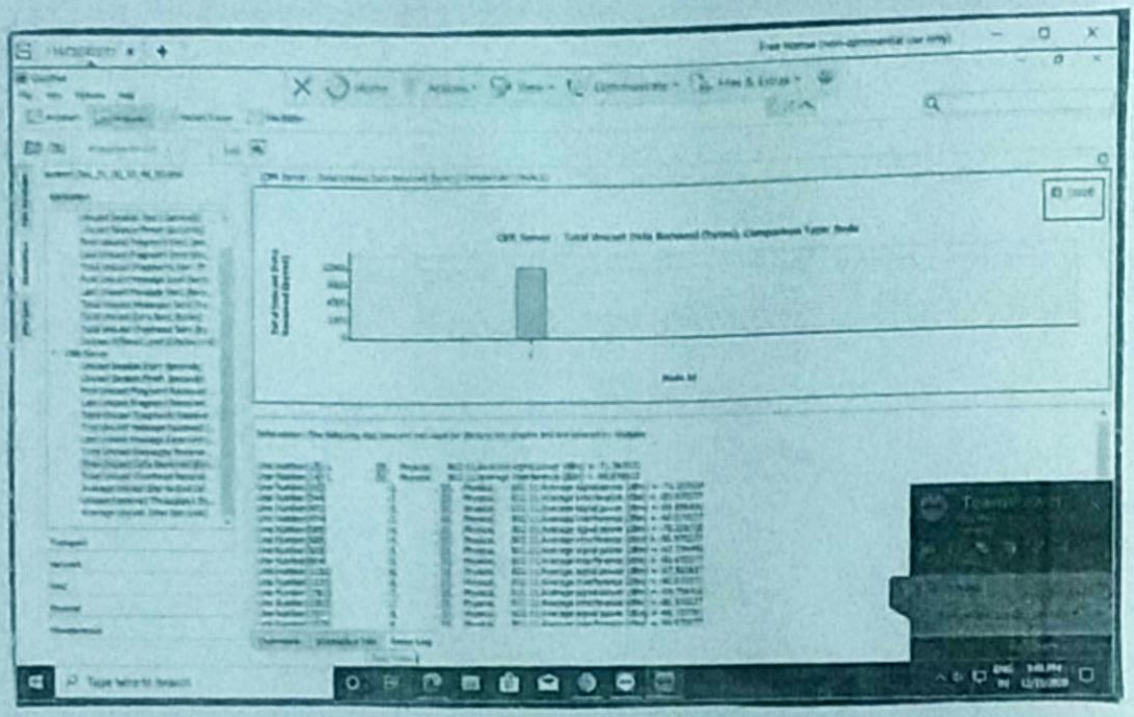


Fig3:

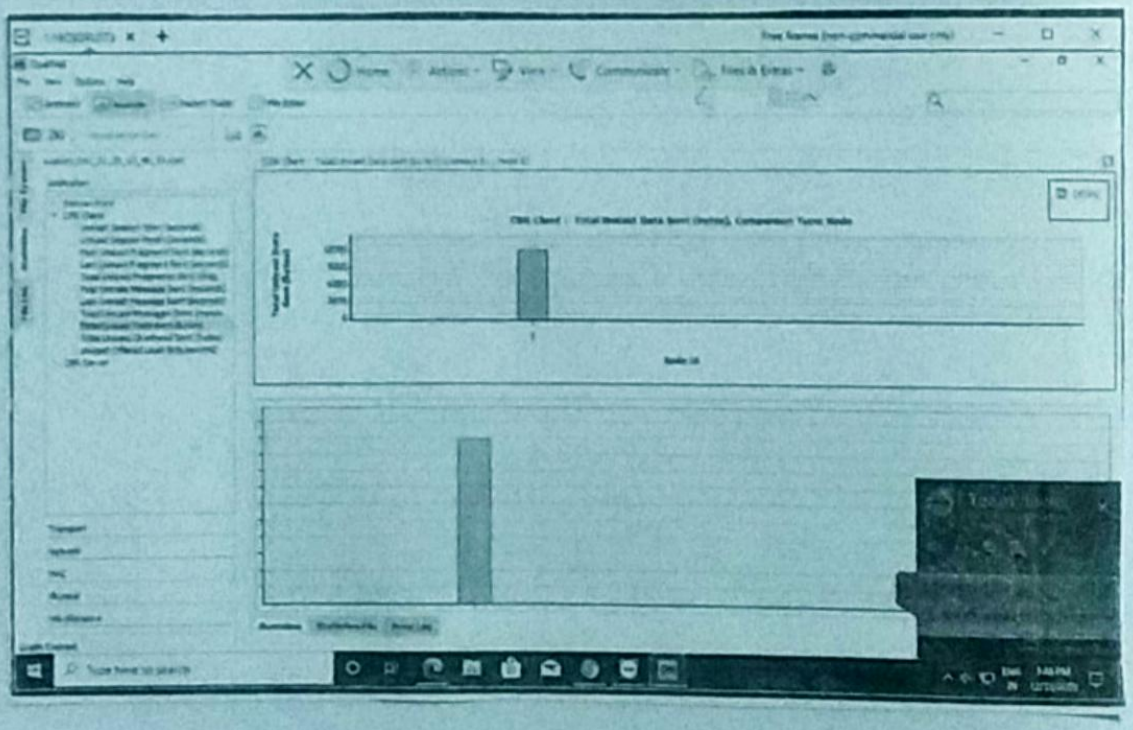


Fig4:

Date

Expt. No.

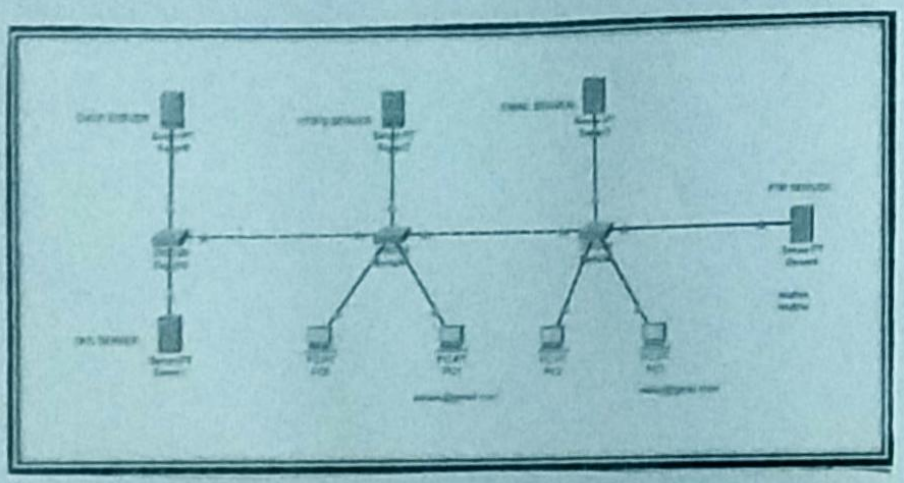
Page No.

Fig3: Total Channel Data Received (bytes) by CBR server

Fig4: Total Channel Data Sent (bytes) by CBR client

Teacher's Signature

Topology A:



Commands used:

DHCP: Subnet Mask
Default Gateway

DNS: Domain Name
IP Address

Email: Domain, Username, Password
Incoming and outgoing Mail servers

FTP: Path ftp.com
ftp>dir

Date

Expt. No.

Page No.

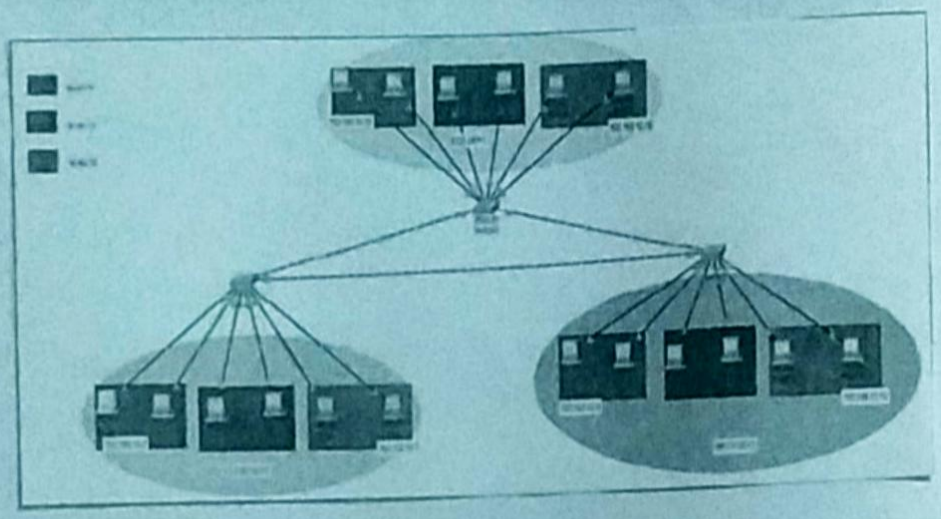
4) Case Study on configuring routers using CISCO packet tracer

Inter VLAN communication

VLAN is a subnetwork which can group together collections of devices on separate physical local area network (LANS). A LAN is a group of computers and devices and share a communication line or wireless link to a server within the same geographical area

Teacher's Signature

Topology B



Date

Expt. No.

Page No.

commands used B

switch > enable

switch # configure terminal

Enter configuration commands, one per line. End with CNTL/Z

switch (config) # vlan 10

switch (config-vlan) # name V10

switch (config-vlan) # exit

switch (config) # vlan 11

switch (config-vlan) # name V11

switch (config) # vlan 12

switch (config-vlan) # name V12

switch (config) # int for 0/1

switch (config-if) # sw m a

switch (config-if) # sw a vlan 10

switch (config) # int fa 0/2

switch (config-if) # sw a vlan 11

switch (config-if) # int fa 0/3

switch (config-if) # sw a vlan 12

~~switch (config-if) # int fa 0/4~~

~~switch (config-if) # int f sw mt~~

switch (config) # int fa 0/4

switch (config-if) # sw mt

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/4, changed state to down

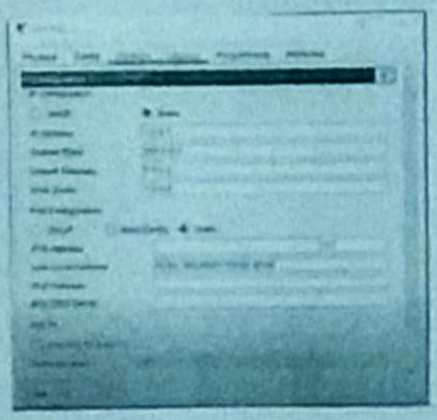
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/4, changed to state up

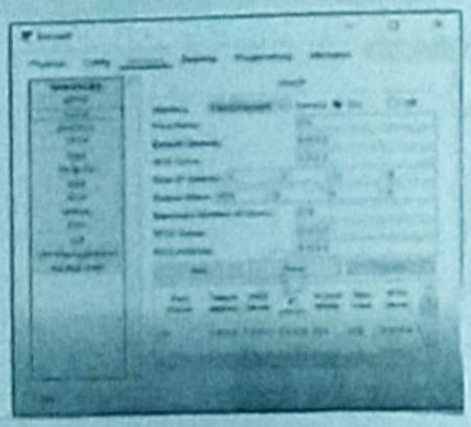
switch (config-if) # sw negotiate

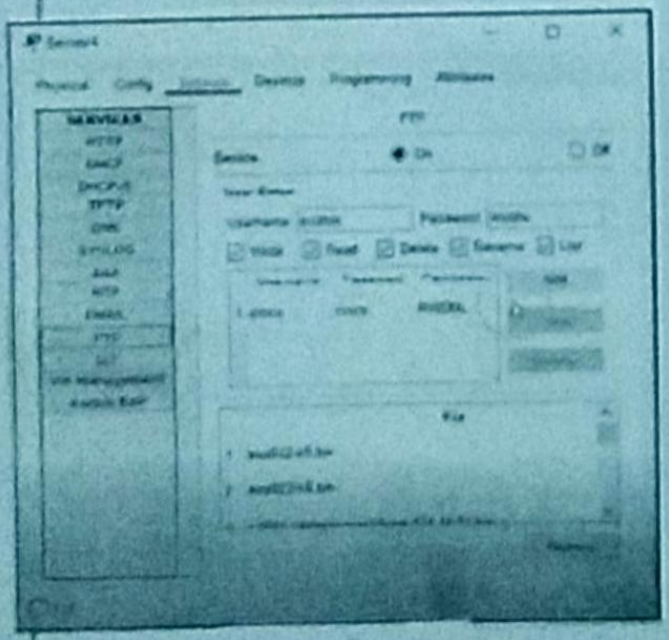
Teacher's Signature

Screenshots of Simulations

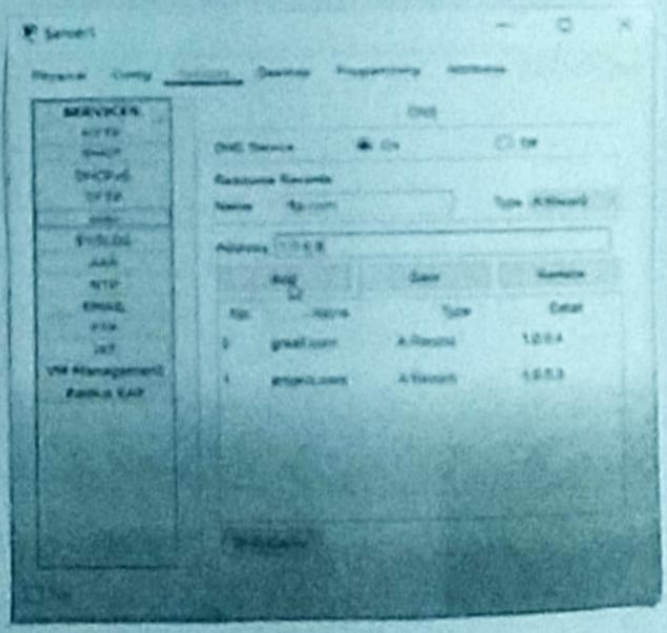
DHCP





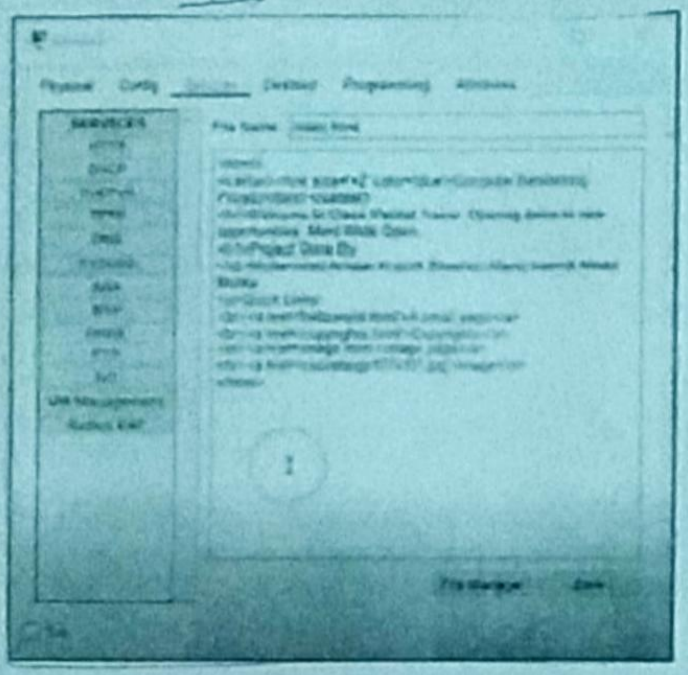


DNS

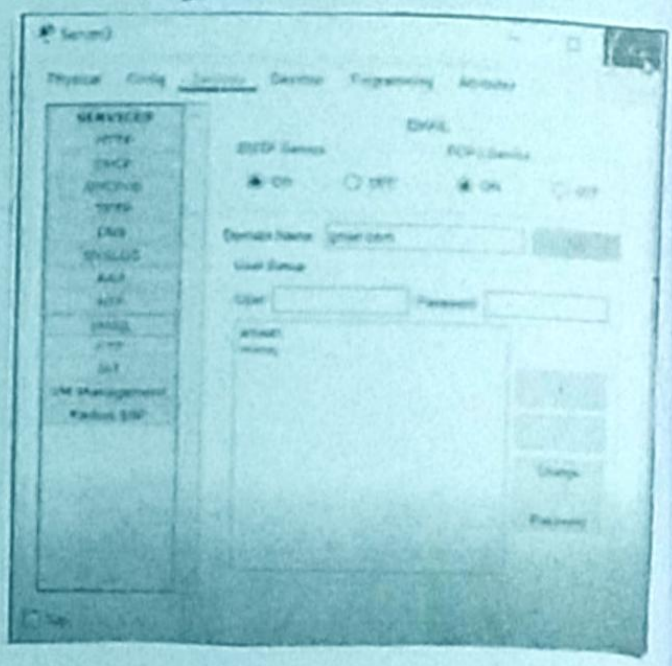


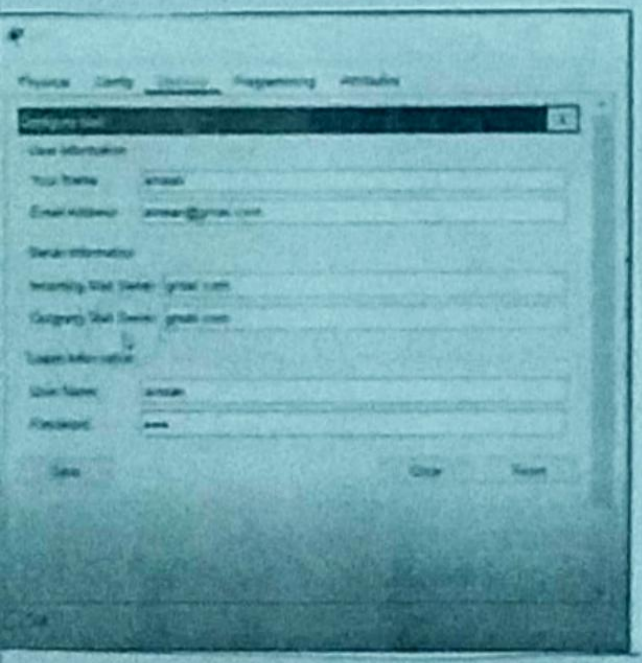
FTP

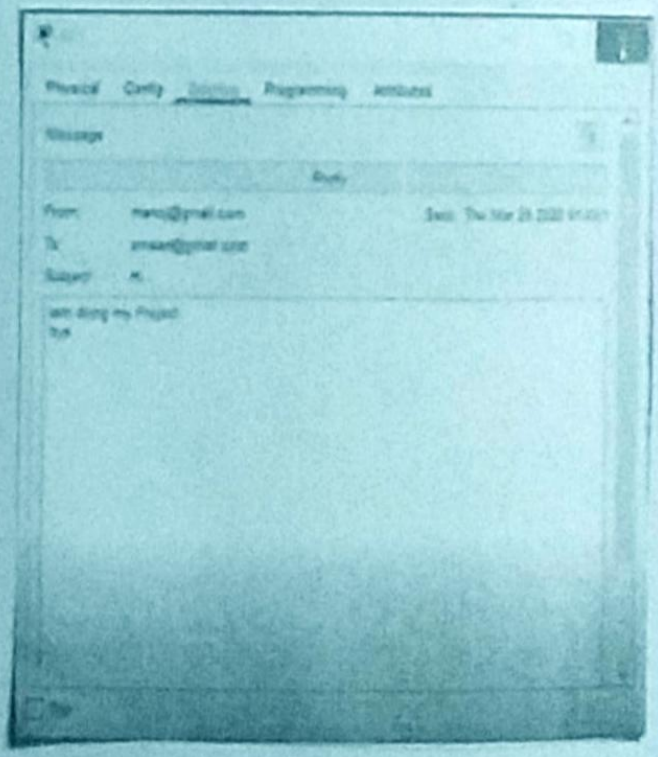
HTTP

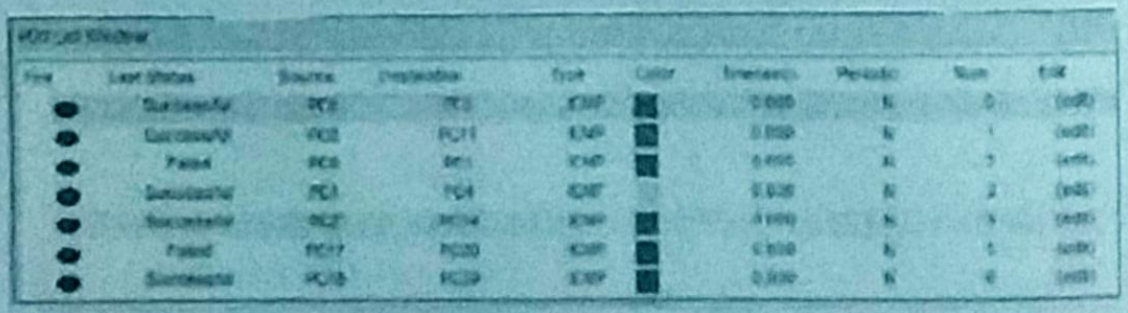


EMAIL





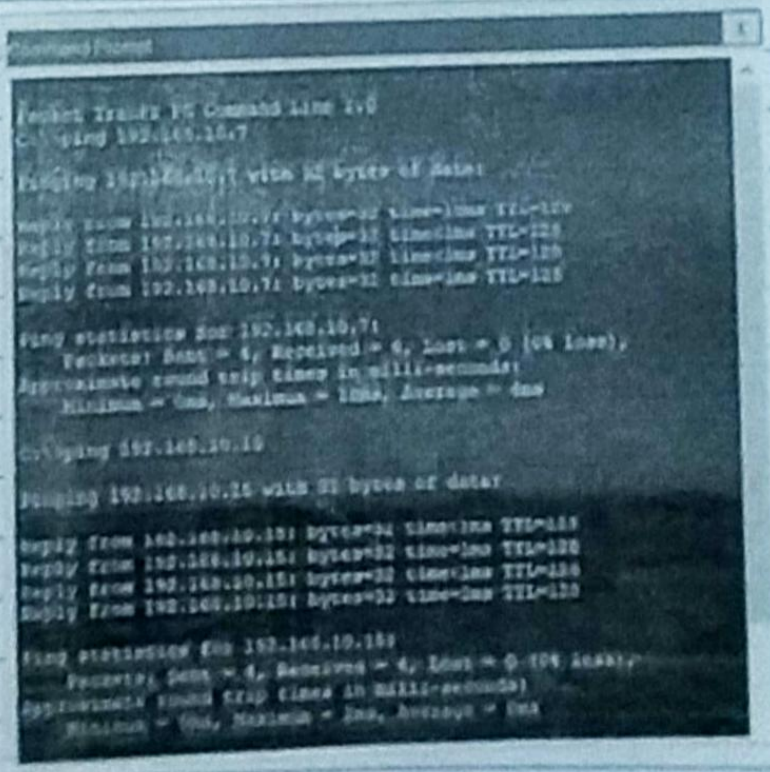




Output

output

```
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/5, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/24
10   V10                              active    Fa0/1, Fa0/6
11   V11                              active    Fa0/2, Fa0/5
12   V12                              active    Fa0/3, Fa0/4
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BridgeMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- ---------- ------ ------
1    enet  100001     1500  -      -      -        -    -          0      0
10   enet  100010     1500  -      -      -        -    -          0      0
11   enet  100011     1500  -      -      -        -    -          0      0
12   enet  100012     1500  -      -      -        -    -          0      0
1002 fddi  101002     1500  -      -      -        -    -          0      0

Switch#
```



Nachiket G Kallapur

USN: 1PI18CS096

Lab Internal

Prog 7

Server.c

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

int main(){
    int sd, acpt, len, bytes, port;
    char send[50], receive[50];
    struct sockaddr_in serv, cli;
    if ((sd=socket(AF_INET, SOCK_STREAM ,0))<0){
        perror("Error in socket\n");
        exit(0);
    }

    bzero(&serv, sizeof(serv));
    serv.sin_family= AF_INET;
    serv.sin_port = htons(8000);
    serv.sin_addr.s_addr= htonl(INADDR_ANY);
    if(bind(sd (struct sockaddr*)&serv, sizeof(serv))<0) {
        fprintf("Error in Bind\n");
        exit(0);
    }
```

```
if (listen(sd,3)<0){
    printf("Error in listen");
    exit(0);
}
if (cfd = accept(sd, (struct sockaddr *)NULL, NULL))<0){
    printf("Error in accept");
    exit(0);
}
while(1){
    bytes = read(cfd, receive, 30, 0);
    receive[bytes]='\0';
    if (strcmp(receive, "end")==0){
        close(cfd);
        close(sd);
        exit(0);
    }
    else {
        printf("Command received : %s", receive);
        system(receive);
        fprintf("\n");
    }
}
}
```

Usint.c

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

int main(){

    int sd, acfd, len, bytes, ford;
    char send[50], receive[50];
    struct sockaddr_in serv, cli;
    if((sd = socket(AF_INET, SOCK_STREAM,0)) <0){
        fprintf("Error in socket\n");
        exit(0);
    }
    bzero(&serv,sizeof(serv));
    serv.sin_family = AF_INET;
    serv.sin_port = Htons(8000);
    serv.sin_addr.s_addr= htonl(INADDR_ANY);
    if(bind(sd, (struct sockaddr*)&serv, sizeof(serv)) <0){
        fprintf("Error in connection\n");
        exit(0);
    }
```

```
while(1){
    printf("Enter the command: ");
    gets(send1);
    if (strcmp(send1, "end") != 0){
        send(sd, send1, 50, 0);
        close(sd);
        break;
    }
  }
}
```

Marks - Part A - Programs

Prog1: 6+4 = 10

Prog2: 6+4 = 10

Prog3: 6+4 = 10

Prog4: 6+4 = 10

Prog5: 6+4 = 10

Prog6: 6+4 = 10

Prog7: 6+4 = 10

Prog8: 6+4 = 10

Total: 80/80 Reduced to 30/30

Part B - 9/10

Internal: 10/10

Total: 49/50