NAME: Srijan Raj

STD: _____ DIV: A ROLL NO.: 71

SUBJECT: NPS LAB

INDEX

SR. NO.	DATE	TITLE	PAGE NO.	TEACHER'S SIGN
1		LAB1: Introduction to Socket Programming	1-15	
2		LAB2: File Operation using Socket Programming	16-30	
3		LAB3: Chat Server using Socket Programming	31-38	
4		LAB4: Data base operations and DNS using Socket programming	39-45	
5		LAB5: Multiple Clients using Socket programming	46-51	
6.		LAB7: Application development using Socket Programming	52-60	
7		LAB8: Prototyping Network Model using Packet Tracer	61-64	
8		LAB9: Basic Topologies using Packet Tracer	65-68	
9		LAB10: Routing protocol using Packet tracer	69-73	
10		LAB11: DHCP and NAT using Packet tracer	74-77	
11		LAB 12: Wireless Topology and VOIP using Packet Tracer	78-81	

LAB - 1

Introduction to Socket Programming

Q1) Write two seperate C programs (one for server and other for client) using socket APIs for TCP, to implement the client-server model such that the client should send a set of integers along with a choice to search for a number or sort the given set to odd and even to the server. The server performs the relevent operations according to the choice. Client should continue to send the messages until the user enters selects the choice "exit".

server

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

int main()
{
    int size;
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in server_addr, c_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(myport);
```

```
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
memset ( server_addr.sin_zero, '\0', sizeof (server_
 addr.sin_zero) );

bind (sockfd, (struct sockaddr *) &server_addr,
      sizeof (server_addr));

  listen (sockfd, 10);
  printf ("The server is ready for listening.\n");

  size = sizeof (struct sockaddr);

  int afd = accept (sockfd, (struct sockaddr *)&
                    c_addr, &size);

  int buffer [10];

  int choice, sz, temp, ansIndex;
  int odd [10] = {0};
  int even [10] = {0};
  int a=0, b=0;
  while (1)
  {
      recv (afd, buffer, sizeof (buffer), 0);
      recv (afd, &sz, sizeof (sz), 0);
      recv (afd, &choice, sizeof (choice), 0);
      switch (choice)
      {
        case 1 : for (int i=0; i<sz-1 ; i++)
                 {
                   for( int j=0; j<sz-i-1; j++)
                   {
                     if (buffer [j] > buffer [j+1])
```

```
        {
          temp = buffer[j];
          buffer[j] = buffer[j+1];
          buffer[j+1] = temp;
        }
      }
    }
    send(afd, buffer, 40, 0);
    break;

case 2:  for(int i=0; i<sz-1; i++)
         {
           for(int j=0; j<sz-1-i; j++)
           {
             if(buffer[j] < buffer[j+1])
             {
               temp = buffer[j];
               buffer[j] = buffer[j+1];
               buffer[j+1] = temp;
             }
           }
         }
         send(afd, buffer, 40, 0);
         break;

case 3:  for(int i=0; i<sz; i++){
           if(buffer[i] % 2 == 0)
             even[a++] = buffer[i];
           else
             odd[b++] = buffer[i];
         }
         send(afd, odd, sizeof(odd), 0);
```

```
        send (afd, &b, sizeof(b), 0);
        send (afd, even, sizeof(even), 0);
        send (afd, &a, sizeof(a), 0);
        break;

case 4: recv(afd, &temp, 4, 0);
        ansIndex = -1;
        for (int i=0; i<sz; i++){
            if (temp == buffer[i]){
                ansIndex = i;
            }
        }
        send(afd, &ansIndex, sizeof(ansIndex), 0
        break;

case 5: close(sockfd);
        exit(0);
    }
  }
}
```

Client

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#define myport 1234

int main()
{
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1)
    {
        printf("socket creation error");
        exit(0);
    }
    struct sockaddr_in c_addr;
    c_addr.sin_port = htons(myport);
    c_addr.sin_family = AF_INET;
    c_addr.sin_addr.s_addr = inet_addr("127.0.0.1
    memset(c_addr.sin_zero, '\0', sizeof(c_addr.sin_zer

    int size = sizeof(struct sockaddr);
    connect(sockfd, (struct sockaddr*), &c_addr,
            sizeof(c_addr));
    int buffer[10];
    int choice, sz, temp, ansIndex;
    int odd[10] = {0};
    int even[10] = {0};
    int a=0, b=0;
```

```
while (1)
{
    printf ("\n\n1. sort in ascending order");
    printf ("\n 2. sort in descending order");
    printf ("\n 3. split into odd & even arrays");
    printf ("\n4. Search for an element in the
            array");
    printf ("\n5. Exit");
    printf ("\n Enter your choice: ");
    scanf ("%d", &choice);
    switch (choice)
    {
        case 1: printf ("\n Enter the no of elements in
                        array");
                scanf ("%d", &sz);
                printf ("\n Enter the elements in array:");
                for (int i=0; i<sz; i++)
                    scanf ("%d", &buffer[i]);
                send (sockfd, buffer, sizeof(buffer), 0);
                send (sockfd, &sz, sizeof(sz), 0);
                send (sockfd, &choice, sizeof(choice), 0);
                recv (sockfd, buffer, sizeof(buffer), 0);
                printf ("%d", buffer[i]);

                break;

        case 2: printf ("\n Enter the no of elements in
                        array");
                scanf ("%d", &sz);
                printf ("\n Enter the elements in array");
                for(int i=0; i<sz; i++)
                    scanf ("%d", &buffer[i]);
```

```
send (sockfd, buffer, sizeof (buffer), 0);
send (sockfd, &sz, sizeof (sz), 0);
send (sockfd, &choice, sizeof (choice), 0);
recv (sockfd, buffer, sizeof (buffer), 0);
printf ("\n The array in descending order is");

for (int i=0 ; i<sz ; i++)
    printf ("%d ", buffer[i]);
    break;

case 3 : printf ("\n Enter the no of elements in
          array : ");
         scanf ("%d", &sz);
         printf ("\n Enter the elements in array :");
         for (int i=0; i<sz ; i++)
             scanf ("%d", &buffer[i]);
         send (sockfd, buffer, 40, 0);
         send (sockfd, &sz, 4, 0);
         send (sockfd, &choice, 4, 0);

         recv (sockfd, odd, 40, 0);
         recv (sockfd, &b, 4, 0);
         recv (sockfd, even, 40, 0);
         recv (sockfd, &a, 4, 0);

         printf ("\n The odd elements in the array\n");
         for (int i=0; i< b; i++)
             printf ("%d", odd[i]);
         printf ("\n The even elements in the array\n");
         for (int i=0 ; i<a ; i++)
             printf ("%d", even[i]);
         break;
```

```
case 4:  printf ("\n Enter the number of elements
             in the array:");
         scanf ("%d", &sz);
         printf ("\n Enter the elements in array:");
         for (int i=0 ; i<sz ; i++)
             scanf ("%d", &buffer[i]);

         printf ("Enter the elements to be searched:");
         scanf ("%d", &temp);

         send (sockfd, buffer, sizeof(buffer), 0);
         send (sockfd, &sz, sizeof(sz), 0);
         send (sockfd, &choice, sizeof(choice), 0);

         recv (sockfd, &ansIndex, sizeof(ansIndex), 0);
         if (ansIndex == -1)
             printf ("\n Element not found");

         else
             printf ("\n Element found at index %d",
                     ansIndex);

         break;

case 5:  send (sockfd, buffer, 40, 0);
         send (sockfd, &sz, 4, 0);
         send (sockfd, &choice, 4, 0);
         close (sockfd);
         exit(0);
    }
}
}
```

DIP 9.

Q2) Write two seperate C program (one for server and other for client) using UNIX socket API. for UDP, in which the client accepts a string from the user and sends it to the server. The server will check if the string is palindrome or not and send the result with the length of the string and the number of occurance of each vowel in the string to the client. The client displays the recieved data on the client screen. The process repeats until the user enter the string "Halt". Then both the process terminate.

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
# define myport 1234


int main()
{

    int sockfd;
    sockfd = socket (AF_INET, SOCK_DGRAM, 0);
    struct sockaddr_in server_addr, client_addr;
```

```
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(myport);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
memset(&(server_addr.sin_zero), '\0', sizeof(server_addr.sin_zero));

bind(sockfd, (struct sockaddr *)&server_addr,
     sizeof(struct sockaddr));

printf("waiting to recieve");
printf("\n");

char buffer[20];
int size = sizeof(struct sockaddr);
int len=0, i, j;

while(1)
{
    recvfrom(sockfd, buffer, 20, 0, (struct sockaddr*)
        &client_addr, &size);

    if (strcmp(buffer, "Halt")==0)
    {
        close(sockfd);
        exit(0);
    }
    len = strlen(buffer);
    char revbuff[20];
    int a=0, b=0, l=0, o=0, u=0

    for(i=0, j=len-1; j>=0; j--, i++)
    {
```

```
revbuff[i] = buffer[j];
switch (buffer[i])
{
    case 'a':
    case 'A': a++; break;

    case 'e':
    case 'E': e++; break;

    case 'i':
    case 'I': I++; break;

    case 'o':
    case 'O': o++; break;

    case 'u':
    case 'U': u++; break;
    }
}
revbuff[i] = '\0'
char result[20];

if (strcmp(revbuf, buffer) == 0)
    strcpy(result, "Palindrome");

else
    strcpy(result, "Not a palindrome");

sendto(sockfd, result, sizeof(result), 0, (struct sockaddr
    *)&client_addr, size);
}
close(sockfd);
return 0;
}
```

client

⇒
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#define myport 1234

int main()
{
    int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(myport);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    memset(addr.sin_zero, '\0', sizeof(addr.sin_zero));

    printf("Succesful till here");

    char buffer[20];
    int size = sizeof(struct sockaddr);
    char result[20];
    int len;
    int a=0, e=0, i=0, o=0, u=0;
    while(1)
    {
        printf("\n Enter the string");
        scanf("%s", buffer);
        if(strcmp(buffer, "Halt")==0)
        {
```

```
sendto (sockfd, buffer, 20, 0, (struct sockaddr *)&
        addr, size);
printf ("Closing the socket (client)");
close (sockfd);
exit(0);
}
sentto (sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)
&addr, size);

recvfrom (sockfd, result, sizeof(result), 0, (struct sockaddr*)
& addr, &size);

recvfrom (sockfd, &len, 4, 0, (struct sockaddr *)&addr,
& size);

recvfrom (sockfd, &a, 4, 0, (struct sockaddr*) & addr,
&size);

recvfrom (sockfd, &e, 4, 0, (struct sockaddr *)&addr,
&size);

recvfrom (sockfd, &i, 4, 0, (struct sockaddr *)&addr,
&size);

recvfrom (sockfd, &o, 4, 0, (struct sockaddr *)&addr,
&size);

recvfrom (sockfd, &u, 4, 0, (struct sockaddr *)&addr,
&size);

printf ("Size of the string = %d \n", len);
```

```
printf ("%.s\n", result);
printf ("Vowel count : \n");
printf ("A = %.d \n", a);
printf ("E = %.d \n", e);
printf ("I = %.d \n", i);
printf ("O = %.d \n", o);
printf ("U = %.d \n", u);
}
return 0;
}
```

OUTPUT Q1

SERVER

```
$ gcc server.c -o s
$ ./s
The server is ready for listening
```

CLIENT

```
$ gcc client.c -o c
$ ./c

1 sort in ascending order
2 sort in descending order
3 split into odd & even array
4 Exit

Enter your choice 1

Enter the element in the array :5
```

Enter the elements
45
34
10
2
The array in ascending order is
2 10 12 34 55

Enter your choice : 3
Enter the no of elements in array: 8

Enter the elements in array:
23 34 45 56 67 78 99 1

The odd element in the array
23 45 67 99 1

The even element in the array
34 56 78

Enter your choice: 4

OUTPUT 02

SERVER

```
$ gcc server.c -o s
$ ./s
waiting to recieve
```

CLIENT

```
$ gcc client.c -o c
$ ./c
Enter the string : hello
size of the string = 5
Not a palindrome
Vowel count
A=0  E=1  I=0  O=1  U=0
```

LAB-2

File Operations using Socket Programming

Q1.) Write two seperate C program (one for server and other for client) using UNIX socket API to implement the following: The user at the client ~~side~~ sends "File not present" to the client and terminates. Otherwise the following menu is displayed at the client side

1. Search 2. Replace 3. Reorder 4. Exit

server

```
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#define MAX_LEN 100

void replaceAll(char *str, const char *oldword, const char
    *newword)
{
    char *pos, temp[1000];
    int index = 0;
    int owlen;
    owlen = strlen(oldword);
```

```
while ((pos = strstr(str, oldword)) != NULL)
  {
      strcpy(temp, str);
      index = pos - str;
      str[index] = '\0';
      strcat(str, newword);
      strcat(str, temp + index + owlen);
  }
}

int main()
{
   int s, r, recb, sntb, x;
  int ca;
  printf("INPUT port number:");
  scanf("%d", &x);
  socklen_t len;
  struct sockaddr_in server, client;
   char buff[50];

   s= socket(AF_INET, SOCK_DGRAM, 0);
   if (s==-1)
     {
        printf("\n Socket creation error.");
         exit(0);
     }
   printf("\n Socket created.");

  server.sin_family = AF_INET
  server.sin_port = htons(x);
 server.sin_addr.s_addr = htons 1 (INADDR_ANY);
  len= sizeof(client);
```

```
ca = sizeof(client);

r= bind (s, (struct sockaddr *) &server, sizeof(server));
 if (r==-1)
   {
      printf ("\n Binding error");
       exit(0);
   }
 printf ("\n Socket Binded");

recb= recvfrom(s, buff, sizeof(buff), 0, (struct sockaddr*)
      &client, &ca);

printf (" File Name recieved");
 char fil[50];
 if ( acess (buff, F_OK) != -1){
       strcpy (fil, buff);
       strcpy (buff, "File exists");
 }
 else {
     strcpy (buff, "File does not exist!");
 }

sntb= sendto (s, buff, sizeof(buff), 0, (struct sockaddr*)
      &client, len);

if (sntb==-1)
   {
     printf ("\n Mussage Sending failed");
      close (s);
     exit (0);
 }
```

```
if (strcmp (buff, "File does not exist!") == 0)
    {
        close (s);
        exit(0);
    }
int ch=0;
while (ch != 4) {
  recb = recfrom (s, buff, sizeof (buff), 0, (struct sockaddr
        &client , &ca);

  if (recb == -1)
    {
       printf ("Message recieving failed");
       close(s);
       exit (0);
    }

ch = buffer[0];
int i, n, n1, n2, j;
char str[50], str1[50], str2[50];
char strTempdata [MAX_LEN];
 int noofline =0;
switch (ch)
{
      case 1:    printf ("\n Searching..\n");
                 n = buff[1];
                 for (i=0; i<n; i++)
                     str[i] = buff[i+2];
                 str[n] = '\0';
                 File *fp
                 int line_num = 1
                 int find_result = 0;
                 char temp[512]
```

```
if ((fp= fopen (fil,"r")) = NULL){
  printf ("\n file not found");
   close(s);
   exit (0);
}

while (fgets (temp, 512, fp ) != NULL){
   if ((strstr( temp, str)) != NULL){
        find_result++;
   }
    line_num++;
}
 if (fp) {
   fclose (fp);
}
 buff [0] = find_result;
 snth = sendto (s, buff, sizeof (buff), 0, (struct
    sockaddr*) &client, len);

if (snth == -1)
{
   printf ("\n Message sending failed");
    close(s);
    exit(0);
}
  break;

case 2:     n1 = buff[1];
            i=2;
          for (j=0; j<n1; j++)
          {
                 str1[j] = buff[i];
```

```
            i++;
        }
        str1[j] = '\0';

    recb = recvfrom(s, buff, sizeof(buff), 0, (struct
            sockaddr *)&client, &ca);

    if (recb == -1)
    {
        printf("\n Message receiving failed");
        close(s);
        exit(0);
    }
        n = buff[2];
        i=2;
        for(j=0; j<n; j++)
        {
            str2[j] = buff[i];
            i++;
        }
        str2[j] = '\0';
        printf("\n Replacing %s with %s.. \n", str1,
                str2);
        FILE *fptr;
        FILE *fTemp;
        char buffer[1000];

        fptr = fopen(fil, "r");
        ftemp = fopen("replace.temp", "w");
        if(fptr == NULL || ftemp == NULL)
        {
            printf("\n Unable to open file.\n");
            exit(0);
        }
```

```
while ((fgets (buffer, 1000, fptr)) != NULL)
{
  replaceAll (buffer, str1, str2);
  fputs (buffer, ftemp);
}

fclose (fptr);
fclose (ftemp);

remove (fil);

rename ("replace.tmp", fil);
strcpy (buff, "Replace finished");
snth = sendto (s, buff, sizeof (buff), 0, (struct
  sockaddr *) &client, len);

if (snth == -1)
{
    printf ("\n Message Sending failed");
    close (s);
    exit(0);
}
  break;

case 3:     printf ("\n Ordering file. \n");

    File * ptrFilelog = NULL;
    fILE * ptrSummary = NULL;

  if ((ptrFilelog = fopen (fil, "r")) = NULL){
      fprint (stderr, "Error : Could not open %s",
                fil);
      return 1;
```

```
while (fgets (strTempData, Max_Len, ptrFilelog) != NULL)
{
    if (strchr (strTempData, '\n');
      strTempData [strLen (strTempData) - 1] = '\0';
     strData = (char**) realloc (strData, sizeof (char**)*
          (noOfLines + 1));
     strData [no of Lines] = (char*) calloc (MAX_LEN,
                  size of (char));
     strcpy (strData [noOfLines], strTempdata);
     no of Lines ++;
    }

for (i=0; i<noofLines; i++)
     fprint (ptrSummary, "%.s", strData[i]);

for (i=0; i< noofLino; i++)
     free (strData[i]);

free (strData);
remove (fil);
rename ("temp. txt", fil);
 fclose (ptrFilelog);
 fclose (ptrSummary);
     strcpy (buff, "Ordering done!");
     Snth = sendto (s, buff, sizeof (buff), 0, (struct
       sockaddr*) &client, len);
if (snth == -1)
   {
       printf ("\n Message sending failed");
        close (s);
        exit (0);
   }
```

```
        break;

    case 4 : ch=4;
        break;
    }
}
close(s);
}
```

OUTPUT

File name : 1.txt
Choice : 2
Replace with we

Initial String	Final String
I am Srijan	we am srijan
I am a student of IT	we am a student of IT

client

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

int main()
  {
    int s, r, recb, sntb, x;
    int sa;
    socklen_t len;
    printf("INPUT PORT NUMBER");
    scanf("%d", &x);
    struct sockaddr_in server, client;
    char buff[50];
    s = socket(AF_INET, SOCK_DGRAM, 0);
    if (s==-1)
      {
        printf("\n Socket Creation error.");
        exit(0);
      }
    printf("\n Socket created");
    server.sin_family = AF_INET;
    server.sin_port = htons(x);
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    sa = sizeof(server);
    len = sizeof(server);
```

```
printf ("Type file name");
scanf ("%s", buff);
sntb = sendto (s, buff, sizeof (buff), 0, (struct sockaddr*)
       &server, len);

if (sntb == -1)
  {
     close (s);
     printf ("\n Message receiving failed");
     exit (0);
  }
recb = recvfrom (s, buff, sizeof (buff), 0, (struct sockaddr*
       &server, &sa);

~~case 3:~~
printf ("\n");
printf ("%s", buff);
printf ("\n\n");

if (strcmp (buff, "File does not exist!") == 0)
  {
     close (s);
     exit (0);
  }

int ch = 0
while (ch != 4)
  {
     printf ("1. serach \n 2. Replace \n 3. Reorder \n 4. Exit
          \n Enter your choice");
```

```
scanf("%d", &ch);
buff[0] = ch

char str1[50], str2[50];
int n, i, j;
switch (ch)
{
 case 1 : printf ("Enter the string to be searched");
          scanf ("%s", str1);
          n = strlen (str1);
          buff[1] = n;
            for (i=0; i<n; i++)
                buff[i+2] = str1[i];
          buff[i+2] = '\0';
          sntb = sendto (s, buff, sizeof(buff), 0, (struct
           sockadd*) &server, len);

          recb = recvfrom (s, buff, sizeof(buff, 0, (struct
           sockaddr*) &server, &sa);

          n = buff[0];
          printf ("\n Word found %d number of times"
                  ,n);
          break;

case 2 :  printf ("Enter the string to be searched
            and replaced :");
          scanf ("%s", str1);
          n = strlen (str1);
          buff[1] = n;
```

```
for(i=0; i<n; i++)
    buff[i+2] = str1[i];
buff[i+2] = '\0';

sntb = sendto(s, buff, sizeof(buff), 0, struct(
        sockaddr *) &server, &sa);

printf("Enter new string");
scanf("%.s", str2);
n = strlen(str2);
~~n = str~~
buff[1] = n;
i = 2;
for(j=0; j<n; j++)
{
    buff[i] = str2[j];
    i++;
}

buff[i] = '\0'

sntb = sendto(s, buff, sizeof(buff), 0, (struct sockaddr*)
        &server, len);

recb = recvfrom(s, buff, sizeof(buff), 0(struct sockadd
        *) &serve, &sa);

printf("%.s\n", buff);
break;

case 3: sntb = sendto(s, buff, sizeof(buff), 0, (struct
        sockaddr *) &server, len);
```

```
recb: recvfrom(s, buff, sizeof(buffer), 0, (struct sock
      addr *) &server, &sa);

      printf("%s \n" buff);
      break;

case 4: sint = sendto(s, buff, sizeof(buff), 0, struct
          (sockaddr *) &server, len);
        if (snth == -1)
        {
            close(s);
            printf("\n Message sending failed");
            exit(0);
        }
        break;
        default: printf("\n Try Again \n");
    }
}
close(s);
}
```

LAB 3 : CHAT SERVER USING SOCKET PROGRAMMING

Q Write a C program using unix server socket API's illustrate full duplex mode chat application between a single client and server using connection oriented service. Display PID and PPID of both parent and child processes

server

```
int main(){

    afd = accept(sockfd, (struct (sockaddr*) &client,
                  &len);

    int pid, ppid;
    pid = fork();
    while (1):
     {
       if (pid==0) {
            recv(afd, buffer, sizeof(buffer), 0);
            if (strcmp (buffer, "BYE")==0).
                 break;
            printf ("%.s", buffer);
       }
      else {
            fget(res, 255, stdin);
            send(afd, res, sizeof(res), 0);
            if (strcmp(res, "BYE")==0)
                 break;
       }
      }
```

PID
PPID
Display

```
close (afd);
close (sockfd);
}
```

CLIENT

```
int main(){

  char msg[30];
  while(1){
    printf ("Enter message");
    scanf ("%s", msg);
    send (sockfd, msg, sizeof(msg), 0);
    recv (sockfd, msg, sizeof(msg), 0);
    if (strcmp(msg, "bye")==0){
      printf ("Chat ended");
      break;
    }
    else{
      printf ("Server: %s in", msg);
    }
  }
  close (sockfd);
  exit(0);
}
```

OUTPUT

SERVER

```
$gcc server.c -o s
$./s
server is listening...
client: Hello
enter message: hi
```

chat ended
$

CLIENT

```
$ gcc client.c -o c
$ ./c
enter message : hello
server : hi
enter message : bye
chat ended
$
```

Q2) SERVER

```
void swap (char *x , char *y){
 char temp;
 temp = *x;
 *x = *y;
 *y = temp; }

void permutate (char *a, int l, int r){
  int i;
  if (i==r){
   printf ("%s\n", a); }
  else {
  for(i=l , i<=r ; i++){
    swap((a+l), (a+i)){
     permutate (a, l+1 , r));
     swap ((a+l), (a+i)); }}}
```

```
int main(){
~~int a,mc~~
 char msg[30];
 while(1){
  recv(a/d, msg, sizof(msg),0);

  if(strcmp(msg,"bye")==0){
   printf("Chat ended");
   break;}

  else{
   printf("Client : %s\n", msg);
   printf("permutation ", msg);
   n = strlen(msg);
   permutate(~~Enter mess~~ msg, 0, n-1);}

   printf("Enter message");
   scanf("%s", msg);
   send(a/d, msg, sizeof(msg), 0);}
   close(sockfd);
  exit(0);}
```

CLIENT

```
int main(){
 char msg[30];
 while(1){
  printf("Enter message:");
  scanf("%s", msg);
  Send(sockfd, msg, sizeof(msg), 0);
  recv(sockfd, &msg, sizeof(msg),0);
```

```
if (strcmp(msg, "bye") == 0){
  printf ("Chat Ended\n");
  break;
  else {
  printf ("Server: .\s\n", msg); }}
  close (sockfd);
  exit (0);
}
```

OUTPUT

SERVER

```
$gcc server.c -o s
  $./s
  server is listening...
  client : abc
  permutation  abc, acb, bac, bca, cba, cab
  enter message : bye
  chat ended
$
```

CLIENT

```
$ gcc client.c -o c
$ ./c
  enter message : abc
  chat ended
$
```

Q3.6 SERVER

```
int main() {
pid_t pid = fork();
if (pid < 0) {
   perror ("In fork");
    exit (1); }
 char buff [100], num [100], alpha [100];

 if (pid == 0) {
 memset (buff, 0, sizeof (buff));
 recv ( c, buff , sizeof (buff), 0);
 int len = strlen (buff);

 int numbers [in];
 int numbers_count = 0;
 for (int i=0 ; i<len ; i++ ){
   if (isdigit [ buff[i])){
    numbers [number_count++] = buff (i) - '0'; }}

 for (int i=0 ; i< number_count -1 ; i++ ){
   for (int j=0; j< number - count - i - 1 ; j++ ){
     if (numbers [j] > numbers [j+1]) {
       int temp = number [j];
       number [j] = number [j+1];
       number [j+1] = temp; } } }

 memset (buff, 0, sizeof (buff));
 printf

 for (int i=0 ;  i<number - count; i++ ){
```

```
snprintf(buff + strlen(buff), sizeof(buff) - strlen(buff),
    "%d", numbers[i]); }

  send(c, buff, sizeof(buff), 0); }
else{
  memset(buff, 0, sizeof(buff));
  recv(c, buff, sizeof(buff), 0);
  int len = strlen(buff);
  int I = 0;
  while (I < len){
  if (isalpha(buff[I])){
    alpha[I++] = buff[I]; }}
  for(int i=0; i<=I-1; i++){
    for(int j=0; j<I-i-1; j++){
      if(alpha[j] < alpha[j+1]){

        char temp = alpha[j];
        alpha[j] = alpha[j+1];
        alpha[j+1] = temp;
      }
    }
  }
  send(c, buff, sizeof(buff), 0);}
  close(c);
  close(s);
  return(0);
}
```

CLIENT

```
int main(){
```

```
printf ("Enter String");
gets (input, sizeof(input), stdin);
Send (c, input, sizeof(input), 0);
memset (buff, 0, sizeof(buff));

printf ("%s", buff);
recv (c, buff, sizeof (buff), 0);
printf ("%s \n", buff);
close (s);
return (0);
}
```

OUTPUT

SERVER

```
$ gcc server.c -o s
$ ./s
  listening
  Chat ended
$
```

CLIENT

```
$ gcc client.c -o c
$ ./c
  enter string : 321 hello
  child PID : 6083
  Ascending order of nos : 123
  parent PID : 6082
  Descending order of alphabets : o l l h e
$ chat ended
```

LAB 4 : DATA BASE OPERATION AND DNS USING SOCKET PROGRAMMING

Q1) SERVER

```
void registration (int c, int childfid){
  char name[] = "Srijan";
  char address[] = "Patna";
  char response[1024];

  snprintf (response, 1024, "%.s \n %.s \n %d", name,
        address, childfid );

  send (c, response, 1024, 0); }

void studentName ( int c, int childfid ){
   char dept[] = "IT";
   cha sem[] = "V";
   cha sec[] = "A";
   char course[] = "NPS".
   char response[1024];
  snprintf (response, 1024; "%.s\n %.s \n%.s \n%.s \n
        %.d", dept, sem, sec, course, childfid );
  send (c, response, 1024, 0);
}

void subjectcode (int c, int childfid){
  char subject code[] = '3161';
   char marks[] = "80%";
   cha response[1024];
  serprintf (response, 1024 "%.s \n %.s \n %.d", subjectcode
      maks, childfid );
```

```
send (c, response, 1024, 0); }

int main()
{
  char buff[1024];

while (1){
  c = accept (s, (struct sockaddr *) &cad, &clen);
  pid_t * childpid = fork();

  if (childpid == 0){
    memset (buffer, 0, sizeof (buffer));

  recv ( c, buff, sizeof(buff), 0);
  int option = stoi (buff);
  switch (option){

  case 1 : registration (c. getpid());
  case 2 : studentName ( c. getpid());
  case 3 : subcode ( c. getpid());
  default : printf ("invalid option \n");
            close (c);
          exit (0); }

  else if ( childpid > 0){
        close (c);
  }
}

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

CLIENT

```
int main()
{
  char buff[1024];
  printf("SELECT \n 1 REG NO \n 2. STUDENT NO \n
      3 SUBJECT CODE \n");

  while (1){
  printf("Enter choice ");
  scanf("%d", &option);
  fflush(stdin);
   sprintf(buff, 1024, "%d", option);
   send(c, buff, 1024, 0);
   memset(buff, 0, 1024);

   r= recv(c, buff, sizeof(buff), 0);
   if(r<0){
    break;}
   printf("Server response: \n%s \n", buff);
  }
  close(c);
  return 0;
}
```

OUTPUT

SERVER

```
$ gcc s.c -o s
$ ./s
  listening...
  chat ended
$
```

CLIENT

```
$gcc c.c -o c
$./c
 MENU
1. REG NO
2. STUDENT NAME
3. SUBJECT CODE

Choice : 1 :
Name : Srijan
Address : Patna
Child PID : 6061
Choice 3
subject code : 3461
marks : 80%.
Childpid : 6062
Choice : 2
dept : IT
sem : V
section : A
course : NPS
Childpid : 6063
Choice 7
invalid
Chat ended
$
```

where address of sub code.

Q2) SERVER

```
int main(){

struct web buff[2];
strcpy(buff[0].domain, "google");
strcpy(buff[0].IP, "111.111.111.0");
strcpy(buff[1],domain, "amaron");
strcpy(buff[1].IP, "0.0.0.1");

while(1){

char msg[15];
recv(afd, msg, sizeof(msg), 0);
if(strcmp(buff[0].ip, msg)==0){

send(afd, buff[0],domain, 15, 0);}
else if (strcmp(buff[1].ip, msg)==0){
send(afd, buff[1], domain, 15, 0);}
else if (strcmp("exit", msg)==0){
  send(afd, "exit", sizeof("exit"), 0);
break;}

else
{
send(afd, "invalid"; sizeof("invalid"), 0);}}
close(5);
exit(0);}
```

CLIENT

```
int main(){
 while (1){
  char msg[15];
  char domain[15];
  printf ("Enter IP");
  scanf ("%s", msg);
  send (s, msg, &b, 0);
  recv (s, domain, 15, 0);
  if (strcmp (domain, "exit")==0){
     printf ("Chat ended");
     break;}
  printf ("domain name %s \n", domain);}
   close (s);
   exit(0);
  }
```

SERVER

```
$ gcc server.c -o s
$ ./s
  listening..
    Chat ended
$
```

CLIENT

```
$ gcc client.c -o c
$./c
   enter IP : 0.0.0.0
   invalid
```

enter IP : 111.111.111.0
google
enter IP : exit
chat ended
$

LAB 5 Multiple Clients Communication using Socket Programming

Q1. SERVER

```
int main(){
if (no_of_client ==2){
   close (ns);
  break; }

no_of_clients++;

if ((Childpid = fork() ==0){
  close (s);
  recv (ns, buff, sizeof(buff),0);
  FILE *fptr;

  fptr = fopen ("temp.txt"; "a");
  fputs ( buff, fptr);

  fclose (fptr);
  char buff2[50];
  recv (ns, buff2, sizeof (buff2), 0);
  FILE * fptr2;

  fptr2 = fopen ("temp2.txt', 'a");
  ft
  fputs (buff2, fptr2);
  f close (fptr2);

  if (no_of_clients == 2){
    FILE *fp;
```

```
int line_num = 1;
char temp[512];
if ((fp = fopen("temp.txt", "V")) == NULL){
close(s);
exit(0); }

while (fgets(temp, 512, fp) != NULL){
strcmp(str, temp);
printf("%.s", str);
~~int nbr~~
line_num++;
}
if (fp){
strcat(ip, " ");
printf("\n IP is : %.s", ip);
send(s, ip, sizeof(ip), 0);
close(s);
}
```

OUTPUT

SERVER

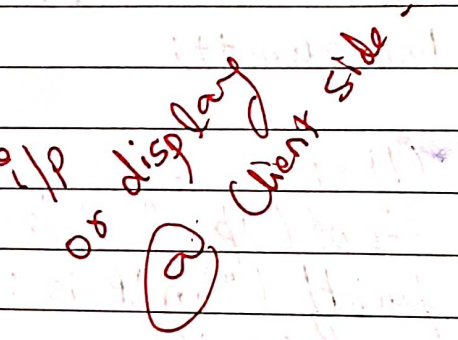
```
$gcc serun.c -o s
$./s
listening...
Manipal Institute of technology
1 ADDRESS 127.0.0.1
2 ADDRESS 127.0.0.1
chat terminated
$
```

CLIENT 1

```
$gcc client1.c -o c1
$./c1
connection established
session terminated
$
```

CLIENT 2

```
$gcc client2.c -o c2
$./c2
connection established
session terminated
$
```



Q2. SERVER

```
int find_anagram(char arr[], char arr2[]){
  int num1[21]={0}, num2[26]={0}, i=0;
  while (arr1[i]!='0'){
  num1[arr1[i]-'a']++;
    i++;}
  i=0;
  while (arr2[i]!='0'){
  num2[arr2[i]-'a']++;
    i++;}
  for (i=0; i<26; i++){
   if (num1[i]!=num2[i])
      return 0;}
      return 1;}
    int num(){
      while (1){

    s = accept(s, (struct sockaddr*) &server, &len);
    r= recv(s, buff, 1024, 0);
    printf("\n Client Socket port number %d \n"
          , x);
    printf("\n :.s", buff1);
    r= recv(s, buff1, sizeof(buff1), 0);
    printf("\n :.s", buff1);
    r= recv(s, buff2, sizeof(buff2), 0);
    printf("\n :.s", buff2);

  if (find_anagram(buff1, buff2)==1){
      strcpy(buff, "Yes");}
    else{
```

```
strcpy (buff, "NO"); }
send (s, buff, 1024, 0);
close (s); }
```

Client

```
int main(){

char ip[50];
strcpy (ip, "127.0.0.1");
send (s, ip, sizeof(ip), 0);
scanf ("%s", buff);
send (s, buff, 1024, 0);
recv (s, buff, sizeof (buff), 0);
printf ("%s\n", buff);
close (s);
}
```

OUTPUT

SERVER

```
$ gcc server.c -o s
$ ./s
1 IP 127.0.0.1
2 IP 127.0.0.1
session terminated
$
```

CLIENT 1

```
$ gcc client.c -o c1
$ ./c1
```

str: listen
Yes, Anagrams
session terminated
$

Client 2

$ gcc client.c -o c2
$ ./c2
str: Silent
Yes, Anagrams
session terminated
$

LAB : 7 Application Development using Socket Programming

LAB EXERCISES

Q1.) "Banking Application" username and password sent to server. Server checks if correct or not. Case Cipher to encrypt password

(a) If incorrect ~~pass~~ username, display Incorrect username

(b) else 'Incorrect password'

On successful login Option
1. delit
2. Credit
3. View balance
4. exit.

→

```
struct users
{
    char username [50];
    char password [50];
    char balance [50];
};

struct temp {
  char temp[50]; };

void split ( char c [100], struct users u[20], int ind)

{
```

```
char delim[] = ",";
int i=0;
char *ptr = strtok(l, delim);

while (ptr != NULL)
{
  strcpy(t[i].temp2, ptr);
    i++;
    ptr= strtok(NULL, delim);
}
strcpy(u[ind].username, t[0].temp2);
strcpy(u[ind].password, t[1].temp2);
strcpy(u[ind].balance, t[2].temp2);
}

int main()
{
FILE *fp;
if ((fp= fopen("users.txt", "r"))==NULL)
{
    printf("Error");
    exit(1);
}
int j=0;
char c[100];
while (fgets(c, sizeof(c), fp))
{
    split(c, u, j);
    j++; }

int s2=j;
fclose(fp);
```

```
len = sizeof (client);
accept (ns, (struct sockaddr *)& client, &len);
ns = accept (s, (struct sockaddr *)& client, &len);

if (ns == -1)
 {
    close (s);
    exit (0);
 }
printf ("Socket Accepting");
int flag = 0;
int usr = 0;

for (i=0 ; i<52; i++){
 if ( strcmp (username, u(i).username) == 0)
    {
        usr = 1;
        strcpy( buf, "success");
        break;
    }
else
   {
    strcpy (buf, "Invalid Password!");
     break;
} } }
if (flag == 0)
    strcpy (buf, "Invalid username");
    send (ns, buf, sizeof (buf), 0);
    if (snth == -1)
    {
        printf ("failed");
          exit(0); }
```

```
while (1){
int m;
recv (ns, &m, 4, 0);
int temp;
switch (m){
case 1:
  temp = atoi (u[curr].balance);
  recv (ns, &flag, 4, 0);
  temp = flag
  sprintf (c, "%.5", temp);
  strcpy ( u[curr].balance, c);
  break;
case 2:
  temp = atoi(u[curr.balance);
  temp += flag;
  sprintf ("c, "%.5", temp);
  strcpy( u[curr].balance, c);
  strcpy (buf, u[cur].balance);
  break;

case 3;
 strcpy (buf, u[curr].balance);
  break;
case 4:
   close (ns);
   close (s);
   exit (0);
   break;
}
send (ns, buff, sizeof (buf), 0);
}
close (s);
}
```

```
void caesarCipher(char *text, int shift)
{
    int i;
    for(i=0; text[i]!='\0'; i++){
        if(isalpha(text[i])){
            char base = islower(text[i]) ;
            text[i] = (text[i] - base + shift) % 26 + base;
    }
    }
}
```

Client

```
int opte;
char res[256];
char username[256];
char password[256];
float amt;
printf("1. Login 2: Exit \n Enter Option");
scanf("%d", &opt);

if(opt==1){
    send(client socket, &opt, sizeof(opt), 0);
    printf("Username: ");
    send("%s", username);
    print("password");
    scanf("%s", password);

send(clientSocket, username, sizeof(username), 0);
send(clientSocket, password, sizeof password), 0);

recv(clientSocket, res, sizeof(res), 0);
```

```
printf("%s", res)
if(strcmp(res, "login successful.") == 0){
  while(1){
    printf("1. Debit 2. Credit 3. Balance 4. Exit");
    printf("\n Enter option");
    scanf("%d", &opt);

    send(clientSocket, opt, sizeof(opt), 0);
    if(opt == 1)
      printf("Enter amt to debit");
      scanf("%f", &amt);
    send(ClientSocket, &amt, sizeof(amt), 0)
    send(ClientSocket, &res, sizeof(res), 0);
    }
}
}
close(s)
return 0;
}
```

O/P

Server

Banking server listening on 127.0.0.1
user Alice logged in
user bob logged in

Client

1. Login 2 Exit
Enter Option 1
Enter username: Alice
Enter password: secret
login successful
1. Debit 2 Credit
Enter option 3
Balance : 30000
Enter Option: 4

Q. user sends filename containing text to server server check file existance. Performs

(1) Child process at server convert text to uppercase

(2) Client reads result from file
(3) Both process write result and PDS
(4) Parent process, each tells converts to text

→ server

```
void convert (char * text ){
 int i=0;
 while (text[i] != '\0'){
  if (text[i]>='a' && text[i]<='z'){
      text[i]= text[i]-32;
 } i++; }}

void sup1(char * text ){
 int i= 0
 while (text[i] != '\0' ){
 if (text[i] >= 'a' && text[i] <= 'z')
      text[i]= text[i] -'a' + 'A';
   } i++; }
}
int main ()
{
 while (1)
 cs = accept (ss, (struct sock addr *)& cli, & l)
 if(cs == -1)
```

```
{ printf("fail");
  close(s);
  exit(0);
}
pid_t cp = fork();
if (cp < 0){
  perror("Socket failed");
  exit(0);
}

if (cp == 0){
  char rec[1024];
  recv(cs, rec, sizeof(rec), 0);
  printf("Recieved : %s\n", rec);
  convert(rec);
  FILE *file = fopen("server_output.txt", "a");
  if (file == NULL){
    printf("failed");
    exit(0); }
  fprintf(file, "Uppercase (child.PID : %d) : %s",
          getpid(), rec);
  fclose(file);
  close(cs);
  exit(0);
}
fprintf(file, "DigitText (Parent_PID : %d) : %s", getpid(),
        rec);
  fclose(file);
  close(cs);
}}
close(ss);
return 0;
}
```

Client

```
char fl[256];
printf("Enter file with text: ");
scanf("%s", fl);
FILE *file = fopen(fl, "r");
if (file == NULL){
    printf("failed");
    exit(0);
}
char fl_data[1024];
fread(fl_data, sizeof(char, sizof(fl_data,
                                  fle);
fclose(file);
send(cs, fl_data, sizof(fl_data), 0);
close(cs);
return 0;
}
```

O/P

Server

Server is listening on 127.0.0.1..
END

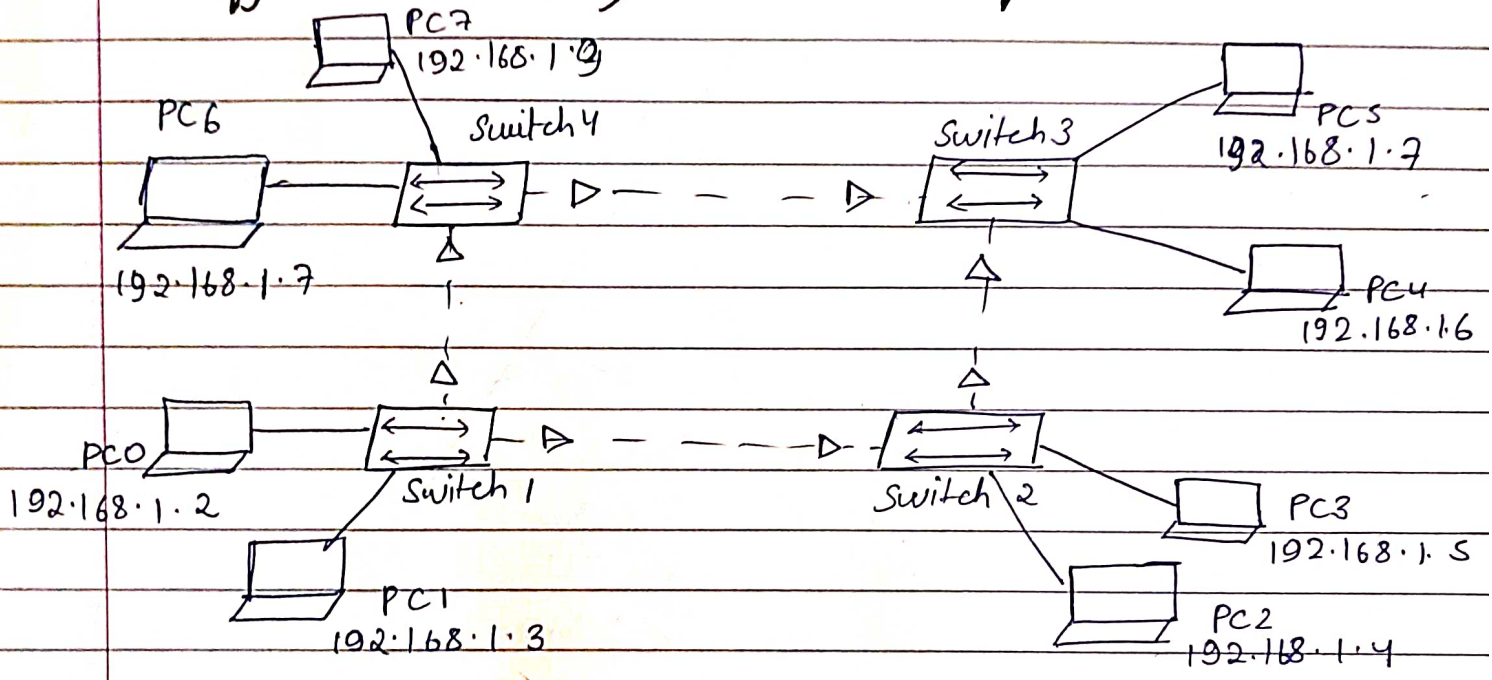
Client

Enter filename : abc.txt
Initial -
hell
final
HELLO
Child process PID: 1011

LAB 8 — Prototyping Network Model using Packet Tracer

LAB EXERCISES:

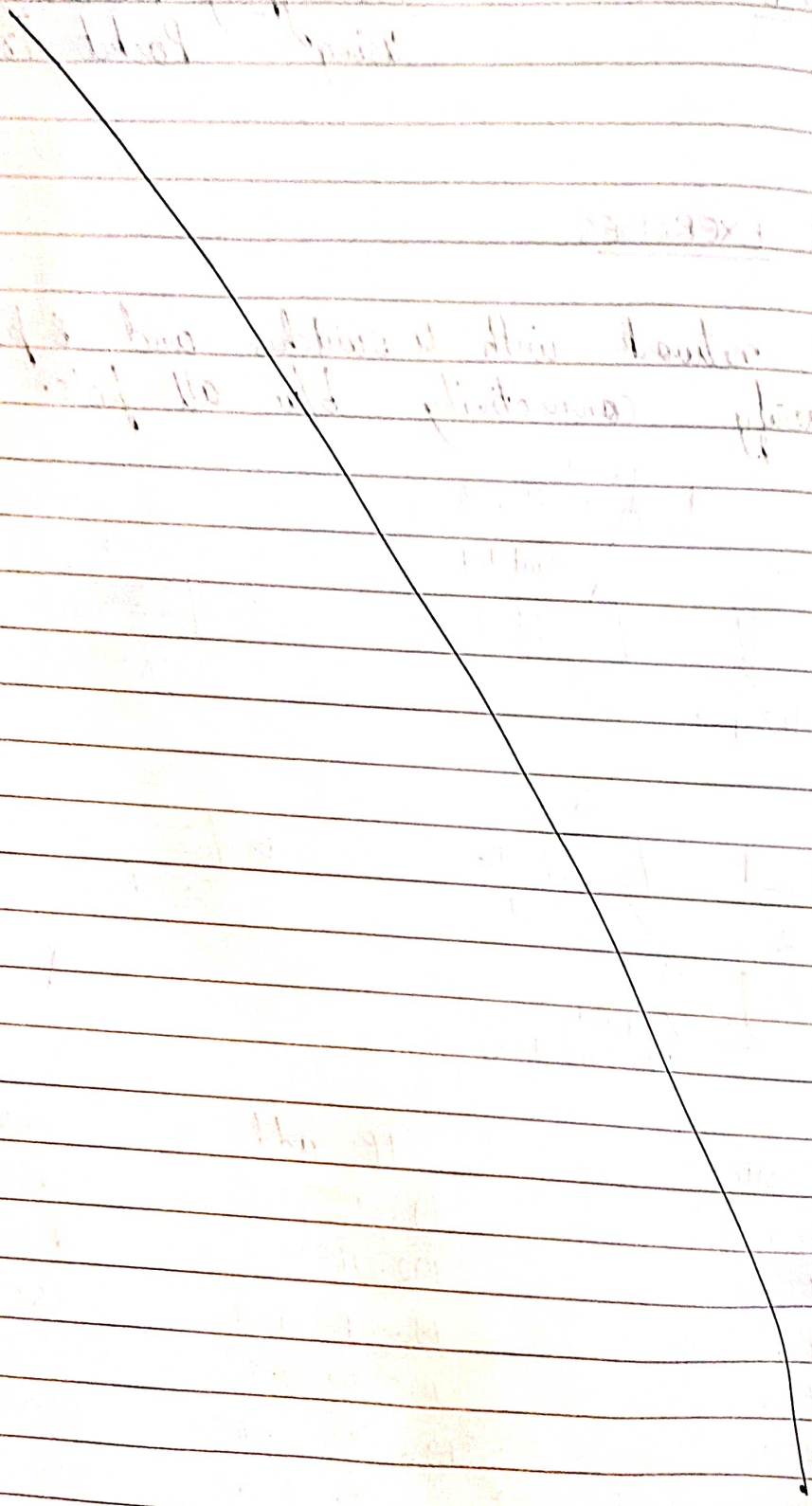
1. Set network with 4 switches and 8 pc's and verify connectivity b/w all pc's.



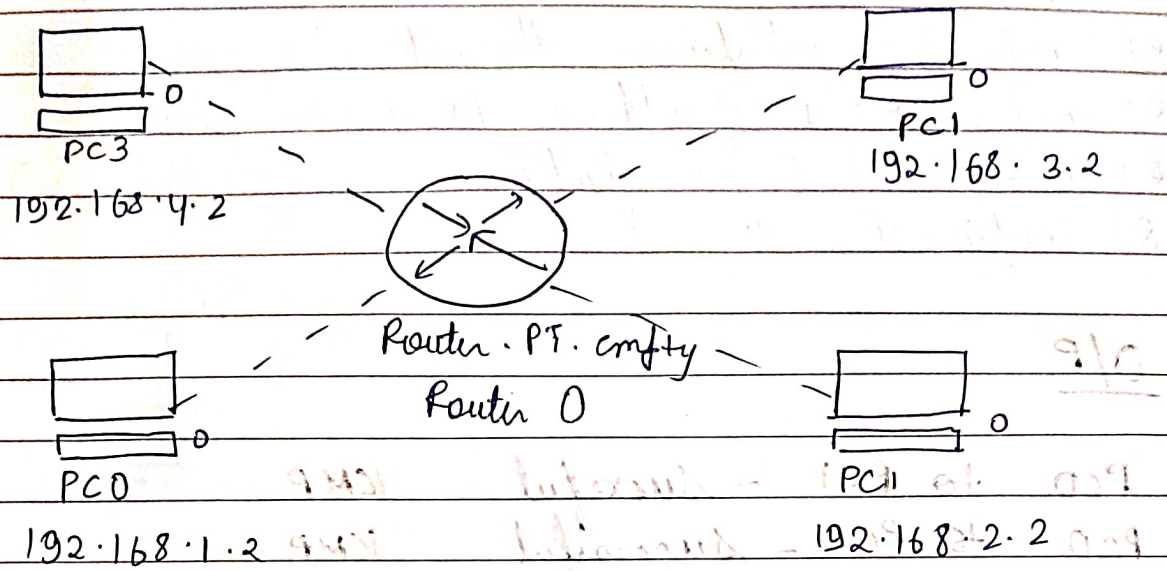
Device	IP add	Subnet mask
PC0	192.168.1.2	255.255.255.0
PC1	192.168.1.3	255.255.255.0
PC2	192.168.1.4	255.255.255.0
PC3	192.168.1.5	"
PC4	192.168.1.6	"
PC5	192.168.1.7	"
PC6	192.168.1.8	"
PC7	192.168.1.9	255.255.255.0

OIP

PC0 to PC6 - Succesful ICMP PC3 to PC1 - Successfull ICMP
PC4 to PC9 - Successfull ICMP PC6 to PC5 - Successful ICMP



Q2. Design and Configure network using 4 PC's and a router. Config done using CLI



Device	IP add	Subnet	Default Gateway
PC0	192.168.1.2	255.255.255.0	192.168.1.1
PC1	192.168.2.2	255.255.255.0	192.168.2.1
PC2	192.168.3.2	255.255.255.0	192.168.3.1
PC3	192.168.4.2	255.255.255.0	192.168.4.1

Router

```
R1 > enable
R1 # conf t
R1 (config) # interface ethernet 0/0
R1 (config-if) # ip address 192.168.1.2 255.255.255.0
R1 (config-if) # no shutdown
R1 (config-if) # exit
R1 (config) # interface ethernet 1/0
R1 (config-if) # ip address 192.168.2.2 255.255.255.0
R1 (config-if) # no shutdown
R1 (config-if) # exit
R1 (config) # interface ethernet 2/0
```

R1 (config-if) # ip address 192.168.3.2 255.255.255.)
R1 (config-if) # no shutdown
R1 (config-if) # exit
R1 (config) # interface ethernet 3/0
R1 (config-if) # ip addrss 192.168.4.2 255.255.23
R1 (config-IF) # no shutdown
R1 (config-if) # exit

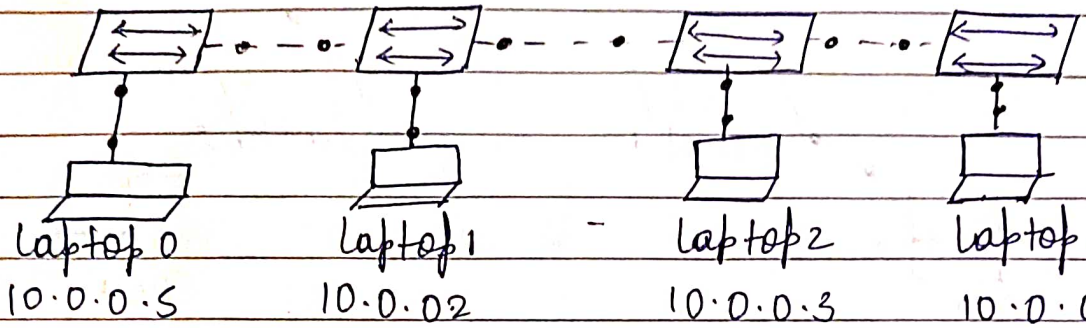
O/P

PC0 to PC1 - Successful ICMP
PC0 to PC2 - Successful ICMP
PC0 to PC3 - Successfull ICMP

LAB:9 Basic Topologies using Packet Tracer

Lab Exercises

1) Bus Topology



Device	IP	Subnet mask
Laptop - PT0	10.0.0.5	255.0.0.0
Laptop - PT1	10.0.0.2	255.0.0.0
Laptop - PT2	10.0.0.3	255.0.0.0
Laptop - PT3	10.0.0.4	255.0.0.0

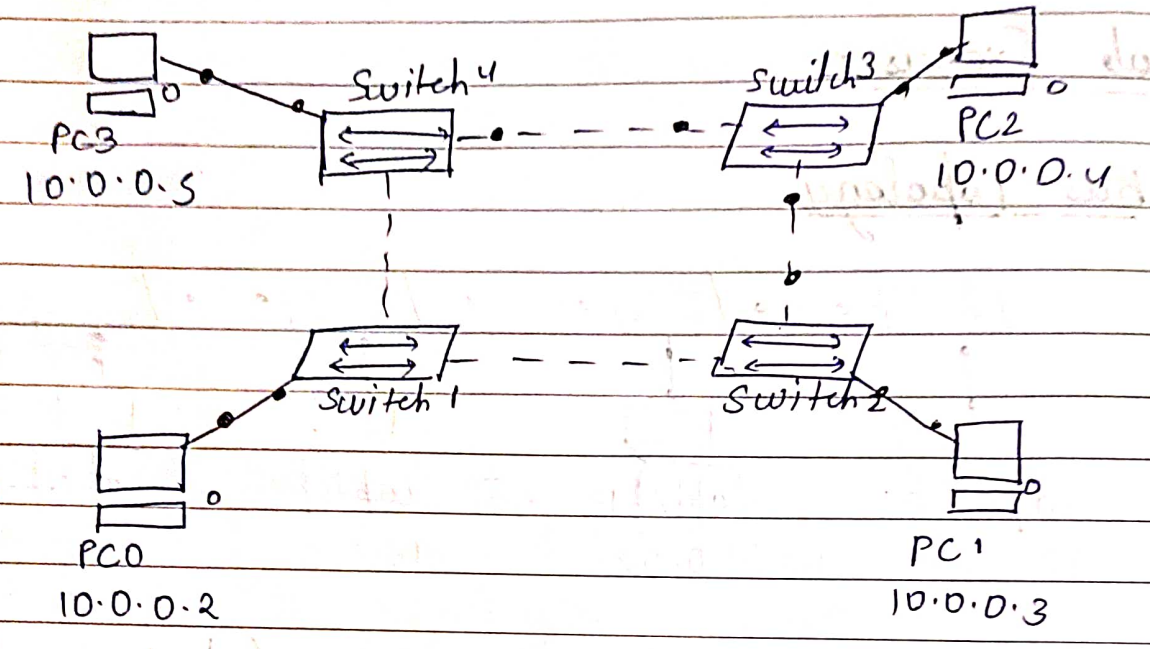
O/P

Laptop 1 to Laptop 2 - Successful ICMP
Laptop 2 to Laptop 3 - Successful ICMP
Laptop 3 to Laptop 4 - Successful ICMP
Laptop 4 to Laptop 1 - Successful ICMP

Laptop 1

C:\> ping 10.0.0.4
pinging 10.0.0.4 with 32 bytes of data
packets: sent: 4 received = 4, loss = 0 (0% loss)

2) Ring topology

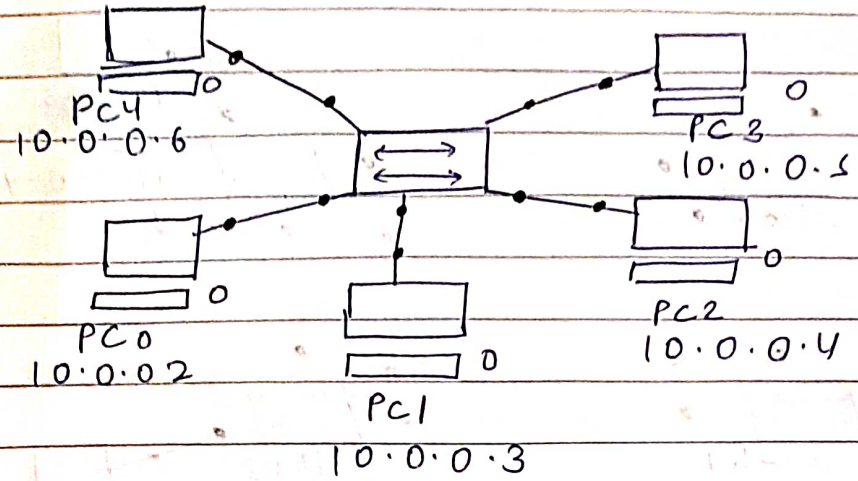


Device	IP	Subnet
PC0	10.0.0.2	255.0.0.0
PC1	10.0.0.3	255.0.0.0
PC2	10.0.0.4	255.0.0.0
PC3	10.0.0.5	255.0.0.0

O/P

PC0 to PC1 - successful ICMP
PC1 to PC2 - successful ICMP
PC2 to PC3 - successful ICMP
PC3 to PC1 - successful ICMP

Star topology

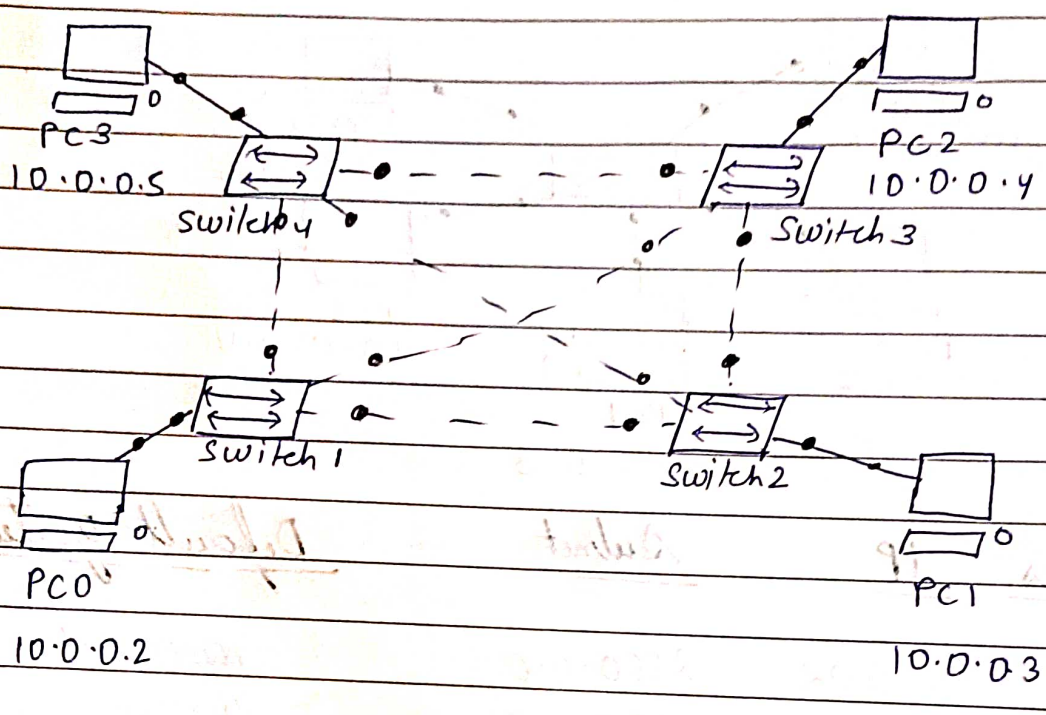


Devices	IP	Subnet	Default gateway
PC0	10.0.0.2	255.0.0.0	10.0.0.1
PC1	10.0.0.3	255.0.0.0	10.0.0.1
PC2	10.0.0.4	255.0.0.0	10.0.0.1
PC3	10.0.0.5	255.0.0.0	10.0.0.1
PC4	10.0.0.6	255.0.0.0	10.0.0.1

O/P

PC0 to PC1 - successful ICMP
PC1 to PC3 - succesful ICMP
PC2 to PC4 - successful ICMP
PC4 to PC2 - Successful ICMP

Mesh Topology



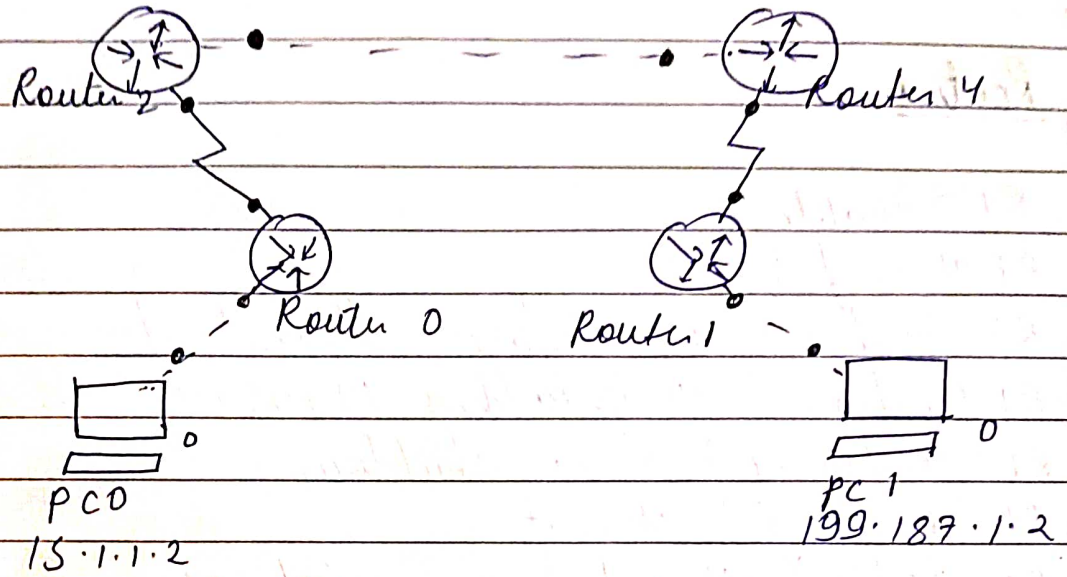
Devices	IP	Subnet
PC0	10.0.0.2	255.0.0.0
PC1	10.0.0.3	255.0.0.0
PC2	10.0.0.4	255.0.0.0
PC3	10.0.0.5	255.0.0.0

O/P

PC0 to PC1 - Successful ICMP
PC1 to PC2 - Successful ICMP
PC2 to PC3 - Successful ICMP

LAB 10 — Configure Routing Protocol using Packet Tracer

1. Using RIP, configure below network and verify connectivity b/w PC0 and PC1



Devices	IP	Subnet	Default Gateway
PC0	15.1.1.2	255.255.255.0	15.1.1.1
PC1	199.18.7.1.2	255.255.255.0	199.187.1.1

Router 0

```
R0 > enable
R0 # conf t
R0 (config) # interface ethernet 0/0
R0 (config-if) # ip address 15.1.1.1 255.255.255.0
R0 (config-if) # no shutdown
R0 (config) # exit
R0 (config) # interface serial 1/0
R0 (config-if) # ip address 10.0.0.2 255.0.0.0
R0 (config-if) # no shutdown
```

```
R0(config-if) # exit
R0(config) # Router RIP
R0(config-router) # network 10.0.0.0
R0(config-router) # exit
```

Router 1

```
R1 > enable
R1 # conf t
R1 (config) # interface ethernet 0/0
R1 (config if) # ip address 199.187.1.2 255.255.255.0
R1 (config-if) # no shutdown
R1 (config-if) # exit
R1 (config) # interface serial 1/0
R1 (config-if) # ip address 12.0.0.2 255.0.0.0
R1 (config-if) # no shutdown
R1 (config-if) # exit
R1 (config) # Router rip
R1 (config router) # network 2.0.0.0
R1 (config-router) # network 199.187.0.0
R1 (config-router) # exit
```

Router 2

```
R2 > enable
R2 # conf t
R2 (config) # interface serial 0/0
R2 (config-if) # ip address 10.0.0.3 255.0.0
R2 (config-if) # no shutdown
R2 (config-if) # exit
```

R2(config) # router rip
R2(config-router) # network 10.0.0.0
R2(config-router) # network 11.0.0.0
R2(config-router) # exit

Router 4

R4 > enable
R4 # conf t
R4(config) # interface serial 0/0
R4(config if) # ip address 11.0.0.3 255.0.0.0
R4(config-if) # no shutdown
R4(config if) # exit
R4(config) # router rip
R4(config-router) # network 11.0.0.0
R4(config-router) # network & 12.0.0.0
R4(config router) # exit

O/P

PC0 to PC1 - succesfull ICMP

Q Do same using OSPF

→ ip address of PC are same

Router 0

R0 > enable
R0 # conf t
R0(config) # interface ethernet 0/0
R0(config if) # ip address 15.1.1.1 255.0.0.0
R0(config-if) # no shutdown

```
R0(config-if) # exit
R0(config) # interface serial 1/0
R0(config-if) # ip address 10.0.02. 255.0.0.0
R0(config-if) # ip ospf 1 area 0
R0(config-if # exit
R0(config) # Router ospf 1
R0(config-router) # network 10.0.0.2
                    0.255.255.255 area 0
R0(config-router)# network 15.1.1.1 0.255.255.255
                    area 0
R0(config-router) # exit
```

Router 1

```
R1 > enable
R1 # conf t
R1 # interface ethernet 0/0
R1(config-if) # ip address 199.187.1.2 255.255.255.0
R1(config-if) # no shutdown
R1(config-if) # exit
R1(config) Router ospf 1
R1(config-router) network 199.187.1.2 0.255.255.255
                    area 0
R1(config-router) network 12.0.0.2 0.255.255.255
                    area 0
R1(config-router) # exit.
```

Router 2

```
R2 > enable
R2 # conft
R2(config) # interface serial 0/0
```

```
R2(config-if)# ip address 10.0.0.3 255.0.0.0
R2(config-if)# ip ospf 1 area 0
R2(config)# interface serial 1/0
R2(config-if)# ip address 11.0.0.2 255.0.0.0
R2(config-if)# ip ospf 1 area 0
R2(config-if)# exit
R2(config)# router ospf 1
R2(config-router)# network 11.0.0.2 0.255.255.255 area 0
R2(config-router)# network 10.0.0.3 0.255.255.255 area 0
R2(config-router)# exit
```

O/P

PC0 to PC1 - successful ICMP

```
C:\>ping 199.187.1.2
Ping 199.187.1.2 with 32 bytes of data
```

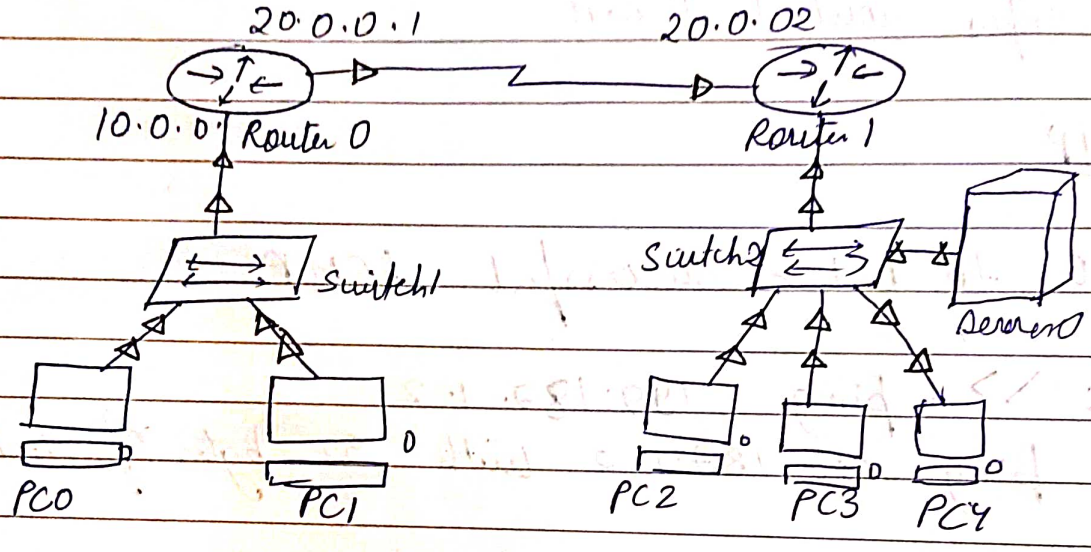
Packets

sent = 4, received = 4 loss = 0 (0% loss)

LAB-11

Configuring DHCP and NAT on Multifunctional device using Packet Tracer

Q1 Configure static NAT for given scenario also ping IP add given and mention interface



Device	IP	Subnet	Default gateway
PC0	10.0.0.2	255.00.0	1.0.0.1
PC1	10.0.0.3	255.0.0.0	1.0.0.1
PC2	10.0.0.2	255.0.0.0	10.0.0.1
PC3	10.0.0.3	255.00.0	10.0.0.1
PC4	10.0.0.4	255.0.0.0	10.0.0.1
Server	10.0.0.5	255.0.0.0	10.0.0.1

Router 0

```
R0> enable
R0 # conf t
R0 (config) # ip nat inside source static 1.0.0.1 2.0.0
R0 (config) # interface gigabitethernet 0/0
```

```
R0(config-if) # ip nat inside
R0(config) # interface serial 0/0
R0(config-if) # ip nat outside
R0(config-if) # exit
R0(config) # exit

R0 # write memory
Building configuration ... [OK]
R0 #
```

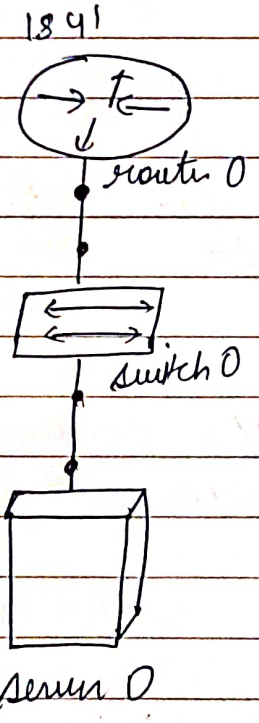
Router 1

```
R1 > enable
R1 # conf t
R1 (config) # ip nat inside source static
R1 (config) # interface gig 0/0
R1 (config-if) # ip nat inside
R1 (config-if) # exit
R1 (config) # inte serial 0/0
R1 (config-if) # ip nat outside
R1 (config-if) # exit
R1 (config) # exit
R1 # wr
Building configuration ... [OK]
R1 #
```

O/P

PC0 to PC2 successful
PC1 to PC3 successful

Q2. Perform NTP client server configuration for below topology



Router 0 → ip → 10.0.0.1
Server 0 → ip → 10.0.0.2

switch 0

switch > enable
switch # conf t
switch (config) # ntp server 10.0.0.2
~~Start~~
switch (config) # clock timezone IST
switch (config) # exit
Switch # show NTP association

address	ref clock	st	when	poll	reach	delay
10.0.0.2	INIT	16	-	64	0	0.00

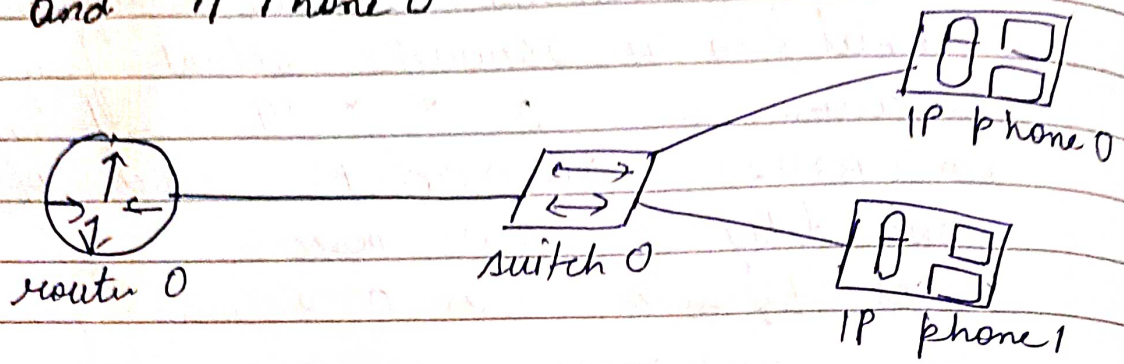
Switch # show clock
5:27:13:65 IST Mon March 1993
switch # clock set 12:00 Nov 2 2023
Switch # show clock
17:0:6:82 IST Thu Nov 2 2023
Switch #

O/P

show ntp status
clock is synchronized, stratum 2, reference is 1.0.0.2
, nomial freq is 25000Hz actual freq 249.990Hz,
precision is 2 ** 19 reference time is
DA50B772. 00 00000F clock offset is 0.000m
root delay is 0.00 msec
root dispersion is 0.02 msec, peer dispersion
is 0.02 msec.

LAB - 12 Wireless topology and voip using packet tracer

Q Configure voip for given topology and IP phone 1 and IP Phone 0



Router 0

```
R0 > enable
R0 # conf t
R0 (config) # int fa 0/0
R0 (config-if) # no shutdown
R0 (config-if) # exit
R0 (config) # ip dhcp pool V1
R0 (dhcp-config) # network 10.0.0.0 255.0.0.0
R0 (dhcp-config) # exit
R0 (config) # telephony-service
R0 (config-telephony) # max-dns
R0 (config-telephony) # max ephones 5
R0 (config-telephony) # ip source address 10.0.0.1
                        port 2000
R0 (config-telephony) # auto assign 4 to 6
R0 (config-telephony) # autoassign 1 to 5
R0 (config-telephony) # exit
R0 (config) # ephone-dn 1
R0 (config-ephone-dn) # number 001
R0 (config-ephone-dn) # exit
```

R0(config)# ephone-dn2
R0(config-ephone-dn)# number 0002

Device	IP	Subnet	default Gateway
IP phone 0	10.0.0.4	255.0.0.0	10.0.0.1
IP phone 1	10.0.0.3	255.0.0.0	10.0.0.1

O/P

Telephone 1 GUI

0002
ringing

pick
connected

Telephone 2 GUI

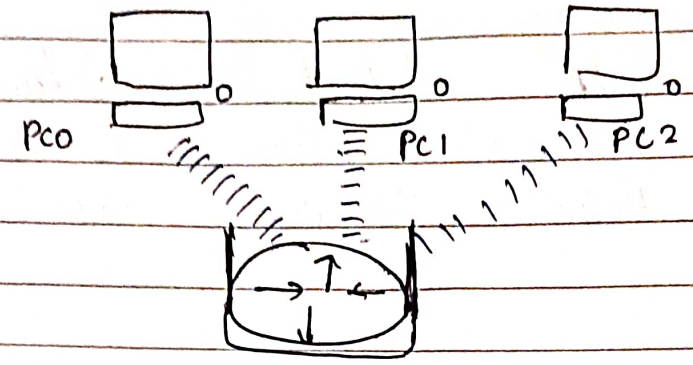
call incoming

0001
dial

O/P

telephone 1 - dial - 0002
Telephone 2 - ring → pitch → connect

Q Linksys wireless router



wireless router 0

Configurations

(a) Set IP for PC's and router using GUI as 192.168.1.2, 192.168.1.3, 192.168.1.4 and 192.168.1.1

(b) Go to router config and set SSID as mother network in GUI under Wireless tab

(c) In router config GUI, select startup, network startup and then change ip address to 10.0.0.1 with mask 255.0.0.0 enable dhcp and start from 10.0.0.2 to automatically assign IP address to PC's and reset the PC.

d) Under wireless tab of config for router and select WAP and then set a hex key of 10 character or more hex: 1234567891

(e) Individually select the PC's and set wireless PC to select SSID as router SSID connect using WAP to set connection.

Device	IP	Subnet	Default Gateway
PC0	10.0.0.2	255.0.0.0	10.0.0.1
PC1	10.0.0.3	255.0.0.0	10.0.0.1
PC2	10.0.0.3	255.0.0.0	10.0.0.1

O/P

PC0 - PC1

packet sent succesfully

C:\> ping 10.0.0.1

ping with 32 bytes of data

packet: sent = 4, recv = 4 loss = 0 (0% loss)