

# LAB RECORD CN CYCLE

## TEST 2 LIKITHA B

1BM19CS079

### CYCLE 2 EXPERIMENTS:

1. Write a program for error detecting code using CRC-CCITT (16-bits).

```
#include
<iostream>
#include
<string.h>

using namespace std;

int crc(char *ip, char *op, char *poly, int mode)
{
    strcpy(op,
ip); if (mode) {
        for (int i = 1; i < strlen(poly);
i++) strcat(op, "0");
        cout << "modified input" << op << endl;
    }
    for (int i = 0; i < strlen(ip);
i++) { if (op[i] == '1') {
        for (int j = 0; j < strlen(poly);
j++) { if (op[i + j] == poly[j])
op[i + j] = '0';          else
        op[i + j] = '1';
        }
    }
    }
    for (int i = 0; i < strlen(op);
i++) if (op[i] == '1')
return 0;    return 1;
}
```

  

```
int main()
{
```

```

char ip[50], op[50], recv[50];
                                cha
r poly[] = "100010000000100001";
                                int
choice;
    cout << "Enter the input message in
binary:"; cin >> ip;
    cout << "generated polynomial is" << poly
<<endl; crc(ip, op, poly, 1);
    cout<<"The checksum is:"<<op+strlen(ip)<<endl;
    cout << "The transmitted message is: " << ip << op + strlen(ip)
<< endl; cout << "do you want to test error" << endl;      cin >>
choice;                                if(choice
==
1)
{
    int
pos,n; char
cp[50];
strcmp(cp,
op);

    cout<<"Enter the position where to insert error
bit"<<endl; cin>>pos;
    cout << "enter bit you wanted to insert"
<<endl; cin >> n;
    cp[pos]=n;
    if(!strcmp(op,
cp))
    {
        cout << "No error"<<endl;
    }
    else
    {
        cout << "Error occured"<<endl;
    }
    return 0;
}
else{ cout << ""<<endl;}
cout << "Enter the received message in binary" <<
endl; cin >> recv;      if (crc(recv, op, poly, 0))
    cout << "No error in data" << endl;
    else

    cout << "Error in data transmission has occurred" << endl;

return 0;
}

```

## OUTPUT :

### Output 1

```
Enter the input message in binary:1011101
generated polynomial is10001000000100001
modified input101110100000000000000000
The checksum is:1000101101011000
The transmitted message is: 10111011000101101011000
do you want to test error
1
Enter the position where to insert error bit
3
enter bit you wanted to insert
0
Error occurred
```

### Output 2

```
Enter the input message in binary:1011101
generated polynomial is10001000000100001
modified input101110100000000000000000
The checksum is:1000101101011000
The transmitted message is: 10111011000101101011000
do you want to test error
0

Enter the received message in binary
10111011000101101011001
Error in data transmission has occurred
```

## 2. Write a program for distance vector algorithm to find suitable path for transmission.

```
#include <bits/stdc++.h>
using namespace std;
#define MAX
10 int n;
class router {
```

```

char adj_new[MAX], adj_old[MAX];
int      table_new[MAX],
table_old[MAX]; public: router( ){
for(int i=0;i<MAX;i++) table_old[i]=table_new[i]=99;
}
void copy( ){ for(int
i=0;i<n;i++) {
adj_old[i]
=adj_new[i];
table_old[i]=table_new[i
];
}
}
int equal( ) {
for(int
i=0;i<n;i++)
if(table_old[i]!=table_new[i]||adj_new[i]!=adj_old[i])return 0; return
1;
}
void input(int j) {
cout<<"Enter 1 if the corresponding router is adjacent to router"
<<(char)('A'+j)<<" else enter 99:
"<<endl<<" "; for(int i=0;i<n;i++) if(i!=j)
cout<<(char)('A'+i)<<" ";
cout<<"\nEnter matrix:";
for(int i=0;i<n;i++) {
if(i==j) table_new[i]=0;
else cin>>table_new[i];
adj_new[i]= (char)('A'+i);
}
cout<<endl;
}
void display(){
cout<<"\nDestination Router: ";
for(int i=0;i<n;i++)
cout<<(char)('A'+i)<<" ";
cout<<"\nOutgoing Line: "; for(int
i=0;i<n;i++) cout<<adj_new[i]<<" ";
cout<<"\nHop Count: ";
for(int i=0;i<n;i++) cout<<table_new[i]<<" ";
}
void build(int j) {
for(int i=0;i<n;i++)
for(int
k=0;(i!=j)&&(k<n);k++
) if(table_old[i]!=99)

```

```

if((table_new[i]+table
-
new[k])<table_new[k
])
{
table_new[k]=table_ne
w[i]+table_new[k];
adj_new[k]=(char)('A'+i);
}
}
} r[MAX]; void
build_table( ) { int
i=0, j=0;
while(i!=n)
{ for(i=j;i<n;i++) {
r[i].copy();
r[i].build(i);
}
for(i=0;i<n;i++) if(!r[i].equal())
{
j=i;
break
;
}
}
}
int main() {
cout<<"Enter the number the routers(<<MAX<<): ";
cin>>n; for(int i=0;i<n;i++) r[i].input(i); build_table(); for(int
i=0;i<n;i++) {
cout<<"Router Table entries for router
"<<(char)('A'+i)<<:"-"; r[i].display();
cout<<endl<<endl;
}
}
}

```

**OUTPUT :**

```

Enter the number the routers(<10): 5
Enter 1 if the corresponding router is adjacent to routerA else enter 99:
  B C D E
Enter matrix:1 1 99 99

Enter 1 if the corresponding router is adjacent to routerB else enter 99:
  A C D E
Enter matrix:1 99 99 99

Enter 1 if the corresponding router is adjacent to routerC else enter 99:
  A B D E
Enter matrix:1 99 1 1

Enter 1 if the corresponding router is adjacent to routerD else enter 99:
  A B C E
Enter matrix:99 99 1 99

Enter 1 if the corresponding router is adjacent to routerE else enter 99:
  A B C D
Enter matrix:99 99 1 99

Router Table entries for router A:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 0 1 1 99 99

Router Table entries for router B:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 1 0 99 99 99

Router Table entries for router C:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 1 99 0 1 1

Router Table entries for router D:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 99 99 1 0 99

Router Table entries for router E:-
Destination Router: A B C D E
Outgoing Line: A B C D E
Hop Count: 99 99 1 99 0

```

**3. Implement Dijkstra's algorithm to compute the shortest path for a given topology.**

```

#include<iostream>
#include<climits>
using namespace
std; int a[30][30],n;
int minimum(int visited[],int dist[])
{
    int mindis=10000,
mini; for(int i=0;i<n;i++)
    {
        if(!visited[i] && dist[i]<mindis)
        {
            mindis=dist[i];
            mini=i;
        }
    }
    return mini;
}

void dijkstra(int src)
{
    int dist[n],visited[n];

    for(int i=0;i<n;i++)
    {
        dist[i]=10000;
        visited[i]=0;
    }
    dist[src]=0;
    for(int i=0;i<n-1;i++)
    {
        int u=minimum(visited,dist);
visited[u]=
1;        for(int v=0;v<n;v++)
        {
            if(!visited[v] && a[u][v]!=10000 && dist[u]!=10000
&& (dist[u]+a[u][v])<dist[v])
                dist[v]=dist[u]+a[u][v];
        }
    }
    cout<<"Shortest paths to all other vertices from "<<src<<" is "<<endl;

```

```

        cout<<"Vertices\tDistance from
source"<<endl; for(int i=0;i<n;i++)
    {
        if(i!=src)
            cout<<i<<"\t\t"<<dist[i]<<endl;
    }
}

int main()
{
    cout<<"Enter the no. of
vertices"<<endl; cin>>n;
    cout<<"Enter the weighted adjacency matrix (enter 10000 if
there is no edge)"<<endl; for(int i=0;i<n;i++)
    {
        for(int
j=0;j<n;j++)
            cin>>a[i][j];
    }
    int src;
    cout<<"Enter the source
vertex"<<endl; cin>>src;
    dijkstra(src);
    return 0;
}

```

## OUTPUT :

```

Enter the no. of vertices
4
Enter the weighted adjacency matrix (enter 10000 if there is no ed
ge)
1 5 7 10000
10000 7 4 2
6 8 0 1
10000 10000 6 3
Enter the source vertex
3
Shortest paths to all other vertices from 3 is
Vertices      Distance from source
0              12
1              14
2              6

```

4. Write a program for congestion control using Leaky bucket algorithm.



```

#include<stdio.h>
>
#include<stdlib.h>
>
#include<unistd.h>

#define NOF_PACKETS
5 int main()
{
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0,
    p_sz, p_time, op;
    for(i = 0; i<NOF_PACKETS; ++i)
        packet_sz[i] = random() % 100;
    for(i = 0; i<NOF_PACKETS; ++i)
        printf("\npacket[%d]:%d bytes\t", i,
        packet_sz[i]); printf("\nEnter the Output rate:");
        scanf("%d", &o_rate); printf("Enter
the Bucket Size:"); scanf("%d", &b_size);
    for(i = 0; i<NOF_PACKETS; ++i)
    {
        if( (packet_sz[i] + p_sz_rm) > b_size)
        if(packet_sz[i] > b_size)/*compare the packet siz with bucket size*/
        printf("\n\nIncoming packet size (%dbytes) is Greater than bucket
capacity (%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
        else
            printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
        else
        {
            p_sz_rm += packet_sz[i];
            printf("\n\nIncoming
Packet size: %d", packet_sz[i]);
            printf("\nBytes
remaining to Transmit: %d", p_sz_rm);
            //p_time = random() * 10;
            //printf("\nTime left for transmission: %d units", p_time);
            //for(clk = 10; clk <= p_time; clk +=
            10) while(p_sz_rm>0)
            {
                sleep(1);
                if(p_sz_rm
                )
                {
                    if(p_sz_rm <= o_rate)/*packet size remaining comparing
with output rate*/
                        op = p_sz_rm, p_sz_rm = 0;
                    else
                        op = o_rate, p_sz_rm -= o_rate;
                }
            }
        }
    }
}

```

```
printf("\nPacket of size %d Transmitted", op);  
    printf(" ---Bytes Remaining to Transmit: %d", p_sz_rm);
```

```
    }  
else  
    {  
        printf("\nNo packets to transmit!!");  
    }  
}  
  
}  
  
}
```

### OUTPUT :

```
packet[0]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:30
Enter the Bucket Size:85

Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 53
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 23
Packet of size 23 Transmitted----Bytes Remaining to Transmit: 0

Incoming packet size (86bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED

Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 47
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 17
Packet of size 17 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 15
Bytes remaining to Transmit: 15
Packet of size 15 Transmitted----Bytes Remaining to Transmit: 0

Incoming packet size (93bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED
```

**5. Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.**

ClientTCP.py

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input("\nEnter file name: ")

clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print("\nFrom Server:\n")
print(filecontents)
clientSocket.close()
```

ServerTCP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file = open(sentence, "r")
    l = file.read(1024)

    connectionSocket.send(l.encode())
    print("\nSent contents of " + sentence)
    file.close()
    connectionSocket.close()
```

**OUTPUT :**

```

C:\Users\Lenovo>cd desktop
C:\Users\Lenovo\Desktop>python tcpserver.py
The server is ready to receive
Sent contents of tcpserver.py
The server is ready to receive

C:\Users\Lenovo\Desktop>python tcpclient.py
Enter file name: tcpserver.py

From Server:

from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file=open(sentence,"r")
    l=file.read(1024)

    connectionSocket.send(l.encode())
    print ('\nSent contents of ' + sentence)
    file.close()
    connectionSocket.close()
C:\Users\Lenovo\Desktop>

```

**6. Using UDP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.**

ClientUDP.py

```

from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET,

```

```

SOCK_DGRAM)
sentence =

```

```

input("\nEnter file name: ")

```

```

clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))

```

```

filecontents,serverAddress =
clientSocket.recvfrom(2048)
print ("\nReply from Server:\n")
print (filecontents.decode("utf-8"))
# for i in filecontents:
#
print(str(i), end = ")")
clientSocket.close()
clientSocket.close()

```

ServerUDP.py

```

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,
SOCK_DGRAM)

```

```

serverSocket.bind(("127.0.0.1",
serverPort)) print ("The server is ready to
receive") while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    l=file.read(2048)

    serverSocket.sendto(bytes(l,"utf-
8"),clientAddress) print ("\nSent contents of '
end = ' ')
print (sentence) # for i in
sentence: # print (str(i), end = "")
            file.close()

```

## OUTPUT :

The screenshot shows two side-by-side Command Prompt windows. The left window, titled 'Command Prompt - python serverudp.py', shows the execution of the server script. It displays the directory path 'C:\Users\Lenovo\cd desktop', the command 'python serverudp.py', and the output 'The server is ready to receive' followed by 'Sent contents of serverudp.py'. The right window, titled 'Command Prompt', shows the execution of the client script. It displays the directory path 'C:\Users\Lenovo\Desktop', the command 'python clientudp.py', and the input 'Enter file name: serverudp.py'. It then shows the output 'Reply from Server:' followed by a large block of Python code that matches the code in the first block. The code in the right window is:

```

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print ("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    l=file.read(2048)

    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)
    print ("\nSent contents of ' , end = ' ' )
    print (sentence)
    # for i in sentence:
    # print (str(i), end = ' ')
    file.close()

```

The right window ends with the prompt 'C:\Users\Lenovo\Desktop>'.