

CN LAB RECORD

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

CODE:

```
#include<bits/stdc++.h>
using namespace std;

string xor1(string a, string b)
{
    string result = "";

    int n = b.length();

    for(int i = 1; i < n; i++)
    {
        if (a[i] == b[i])
            result += "0";
        else
            result += "1";
    }
    return result;
}

string mod2div(string dividend, string divisor)
{
    int pick = divisor.length();

    string tmp = dividend.substr(0, pick);

    int n = dividend.length();

    while (pick < n)
    {
        if (tmp[0] == '1')
            tmp = xor1(divisor, tmp) + dividend[pick];
        else
            tmp = xor1(std::string(pick, '0'), tmp) +
                    dividend[pick];

        pick += 1;
    }
}
```

```

    }

    if (tmp[0] == '1')
        tmp = xor1(divisor, tmp);
    else
        tmp = xor1(std::string(pick, '0'), tmp);

    return tmp;
}

void encodeData(string data, string key)
{
    int l_key = key.length();

    string appended_data = (data +
                             std::string(
                                 l_key - 1, '0'));

    string remainder = mod2div(appended_data, key);
    string codeword = data + remainder;
    cout << "Remainder : "
          << remainder << "\n";
    cout << "Encoded Data (Data + Remainder) : "
          << codeword << "\n";
}

int main()
{
    string dataword;
    cout<<"Enter the dataword"<<endl;
    getline(cin,dataword);
    string key = "10001000000100001";

    encodeData(dataword, key);

    return 0;
}

```

OUTPUT:

```
PS C:\CN Lab> cd "c:\CN Lab\CN-LAB\LAB 7\" ; if ($?) { g++ -O crc *.cpp } ; if ($?) { .\crc }  
Enter the dataword  
110110  
Remainder : 0101011010010101  
Encoded Data (Data + Remainder) :1101100101011010010101  
PS C:\CN Lab\CN-LAB\LAB 7> 
```

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

CODE:

```
#include<stdio.h>
#define inf 999
struct routing{
    int dist[10];
    int hop[10];
};
struct routing nodes[10];

void init(int n){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            if(i!=j){
                nodes[i].dist[j] = inf;
                nodes[i].hop[j] = -20;
            }
            else{
                nodes[i].dist[j] = 0;
                nodes[i].hop[j] = -20;
            }
        }
    }
}

void update(int i,int j,int k){
    nodes[i].hop[j] = k;
    nodes[i].dist[j] = nodes[i].dist[k] + nodes[k].dist[j];
}

void dvr(int n){
    int i,j,k;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            for(k=0;k<n;k++)
                if(nodes[i].dist[j]>(nodes[i].dist[k] + nodes[k].dist[j]))
                    update(i,j,k);
}

int main(){
```

```

int i, j, n;
printf("Enter the number of nodes\n");
scanf("%d",&n);
init(n);
printf("Enter the distance vector\n");
for(i=0;i<n;i++){
    printf("Enter for node %d\n",i);
    for(j=0;j<n;j++){
        scanf("%d",&nodes[i].dist[j]);
    }
}
dvr(n);
printf("\nUpdated distance vector table\n");
for(i=0;i<n;i++){
    printf("Updated node %c table\n",65+i);
    printf("To\t cost\t hop\n");
    for(j=0;j<n;j++){
        printf("%c\t %d\t %c\n",65+j,nodes[i].dist[j],
65+nodes[i].hop[j]);
    }
}

return 0;
}

```

OUTPUT:

```
PS C:\CN Lab\CN-LAB\LAB 8> cd "c:\CN Lab\CN-LAB\LAB 8\" ; if ($?) { g++ -o distance_vector *.cpp } ; if ($?) { .\distance_vector }
Enter the number of nodes
5
Enter the distance vector
Enter for node 0
0
5
2
3
999
Enter for node 1
5
0
4
999
3
Enter for node 2
2
4
0
999
4
Enter for node 3
3
999
999
0
999
Enter for node 4
999
3
4
999
0
```

```
Updated distance vector table
Updated node A table
To      cost  hop
A        0    -
B        5    -
C        2    -
D        3    -
E        6    C
Updated node B table
To      cost  hop
A        5    -
B        0    -
C        4    -
D        8    A
E        3    -
Updated node C table
To      cost  hop
A        2    -
B        4    -
C        0    -
D        5    A
E        4    -
Updated node D table
To      cost  hop
A        3    -
B        8    A
C        5    A
D        0    -
E        9    A
Updated node E table
To      cost  hop
A        6    C
B        3    -
C        4    -
D        9    A
E        0    -
```

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

CODE:

```
#include<bits/stdc++.h>

#include <limits.h>
#include <stdio.h>

using namespace std;

#define V 9

int minDistance(int dist[], bool sptSet[])
{
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

void printSolution(int dist[])
{
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}

void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, sptSet);
```

```

        sptSet[u] = true;

        for (int v = 0; v < V; v++)
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    printSolution(dist);
}

int main()
{
    int graph[V][V] ;
    cout<<"Enter the graph "<<endl;
    for(int i = 0; i<V; i++)
    {
        for(int j = 0; j<V; j++)
            cin>>graph[i][j];
    }

    dijkstra(graph, 0);

    return 0;
}

```


OUTPUT:

Please Enter The Graph (!!! Use 99 for infinity):

0 4 5 99

45 0 2 6

5 8 0 99

4 12 6 0

Enter the source vertex:

0

Vertex	Distance	Path
--------	----------	------

0 -> 1	4	0 1
--------	---	-----

0 -> 2	5	0 2
--------	---	-----

0 -> 3	10	0 1 3
--------	----	-------

PS C:\Users\PUNEETH K\OneDrive\Desktop\LAB_2\CN> █

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

CODE:

```
#include<bits/stdc++.h>
#include<unistd.h>
using namespace std;
#define bucketSize 500

void bucketInput(int a,int b)
{
    if(a > bucketSize)
        cout<<"\n\t\tIncoming packet size "<<a<<" greater than bucket
capacity so-PACKET REJECTED!!";
    else{
        sleep(5);
        while(a > b){
            cout<<"\n\t\t"<<"Packet of size "<<b<<" transmitted"<<"---Bytes
Remaining to transmit= "<<a-b;
            a-=b;
            sleep(5);
        }
        if(a > 0)
            cout<<"\n\t\t"<<"Packet of size "<<a<<" transmitted"<<"---Bytes
Remaining to transmit= "<<0;
    }
}

int main()
{
    int op,pktSize;
    cout<<"Enter output rate : ";
    cin>>op;
    for(int i=1;i<=5;i++)
    {
        sleep(rand()%10);
        pktSize=rand()%700;
        cout<<"\nPacket no "<<i<<"\tPacket size = "<<pktSize;
        bucketInput(pktSize,op);
    }
    cout<<endl;
    return 0;
}
```

OUTPUT:

```
PS C:\CN Lab\CN-LAB\LAB 10> cd "c:\CN Lab\CN-LAB\LAB 10\" ; if ($?) { g++ -o leaky *.cpp } ; if ($?) { .\leaky }
Enter output rate : 100

Packet no 1    Packet size = 267
                Packet of size 100 transmitted---Bytes Remaining to transmit= 167
                Packet of size 100 transmitted---Bytes Remaining to transmit= 67
                Packet of size 67 transmitted---Bytes Remaining to transmit= 0
Packet no 2    Packet size = 600
                Incoming packet size 600 greater than bucket capacity so-PACKET REJECTED!!
Packet no 3    Packet size = 324
                Packet of size 100 transmitted---Bytes Remaining to transmit= 224
                Packet of size 100 transmitted---Bytes Remaining to transmit= 124
                Packet of size 100 transmitted---Bytes Remaining to transmit= 24
                Packet of size 24 transmitted---Bytes Remaining to transmit= 0
Packet no 4    Packet size = 658
                Incoming packet size 658 greater than bucket capacity so-PACKET REJECTED!!
Packet no 5    Packet size = 664
                Incoming packet size 664 greater than bucket capacity so-PACKET REJECTED!!
```

LAB 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.

CODE:

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()
```

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()
```

OUTPUT:

```
Nithin'@NITHS MINGW64 /c/CN Lab/CN-LAB/LAB 11 (main)
$ py clienttcp.py

Enter file name: dummy.txt

From Server:

Hi!! This is a demo file
```

```
Nithin'@NITHS MINGW64 /c/CN Lab/CN-LAB/LAB 11 (main)
$ py servertcp.py

The server is ready to receive

Sent contents of dummy.txt
The server is ready to receive
█
```

LAB 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.

CODE:

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()
```

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()
```

OUTPUT

```
Nithin'@NITHS MINGW64 /c/CN Lab/CN-LAB/LAB 12 (main)
```

```
$ py clientudp.py
```

```
Enter file name: dummy.txt
```

```
Reply from Server:
```

```
Hi!! This is a demo file for udp
```

```
Nithin'@NITHS MINGW64 /c/CN Lab/CN-LAB/LAB 12 (main)
```

```
$ py serverudp.py
```

```
The server is ready to receive
```

```
Sent contents of dummy.txt
```

```
Sent contents of dummy.txt
```

```
□
```

