

## CN LAB REPORT

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

CODE:-

```
Import
java.io.*;

import java.lang.*;
import java.util.*;
class Main
{
    public static String string_val(String sts,int poly_length)
    {
        for(int i=1;i<poly_length;i++)
        {
            sts=sts+"0";
        }
        return sts;
    }

    public static String generate(char[] divisor,char[] dividend,int len,String org)
    {
        for(int i=0;i<len;i++)
        {
            if(dividend[i]=='1')
            {
                for(int j=0;j<divisor.length;j++)
                {
                    if(dividend[i+j]==divisor[j])
                    {
                        dividend[i+j]='0';
                    }
                    else
                    {
                        dividend[i+j]='1';
                    }
                }
            }
        }
        String st=String.valueOf(dividend);

        String fin=org+st.substring(len);
    }
}
```

```

        return fin;
    }
    public static void main(String[] args)
    {

        String str,rec;
        String d="10001000000100001";
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the string");
        str=sc.next();

        String org=str;

        int len=str.length();

        str=string_val(str,d.length());

        char[] divisor=d.toCharArray();
        char[] dividend=str.toCharArray();

        String fin=generate(divisor,dividend,len,org);

        System.out.println("DIVISOR= " + String.valueOf(divisor));
        System.out.println("DIVIDEND= " + String.valueOf(dividend));

        System.out.println("TRANSMITTED MESSAGE IS " + fin);

        System.out.println("Enter the received message");
        rec=sc.next();
        org=rec;

        len=rec.length();

        rec=string_val(rec,d.length());

        dividend=rec.toCharArray();

        String rin=generate(divisor,dividend,len,org);

        System.out.println("MESSAGE DUE TO ERRORS IS");
        System.out.println(rin);
        if(fin.equals(rin))
        {
            System.out.println("NO ERRORS");
        }
        else
        {

```

```
        System.out.println("ERRORS REPORTED");  
    }  
}  
}
```

OUTPUT:-

```
Enter the string  
11111  
DIVISIOR= 10001000000100001  
DIVIDEND= 000001110001111011110  
TRANSMITTED MESSAGE IS 111111110001111011110  
Enter the received message  
1111  
MESSAGE DUE TO ERRORS IS  
11111111000111101111  
ERRORS REPORTED  
  
...Program finished with exit code 0  
Press ENTER to exit console.[]
```

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

CODE:-

```
#include<stdio.h>

struct node
{
    unsigned dist[20];
    unsigned from[20];
}
rt[10];
int main()
{
    int dmat[20][20];
    int n,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&n);
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
        {
            scanf("%d",&dmat[i][j]);
            dmat[i][i]=0;
            rt[i].dist[j]=dmat[i][j];
            rt[i].from[j]=j;
        }
    do
    {
        count=0;
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
```

```

                                for(k=0;k<n;k++)
if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
    {
        rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
        rt[i].from[j]=k;
        count++;
    }
    }while(count!=0);
    for(i=0;i<n;i++)
    {
        printf("\n\nState value for router %d is \n",i+1);
        for(j=0;j<n;j++)
        {
            printf(" \t\nnode %d via %d Distance%d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
        }
    }
    printf("\n\n");
}

```

## OUTPUT:

The image displays two screenshots of a C compiler IDE, likely Visual Studio Code, showing the output of a program. The program appears to be calculating distances between nodes in a network.

**Top Screenshot:**

```
Enter the number of nodes : 4
Enter the cost matrix :
0 3 5 99
3 0 99 1
5 99 0 2
99 1 2 0

State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance3
node 3 via 3 Distance5
node 4 via 2 Distance4

State value for router 2 is
node 1 via 1 Distance3
node 2 via 2 Distance0
node 3 via 4 Distance3
node 4 via 4 Distance1

State value for router 3 is
node 1 via 1 Distance5
node 2 via 4 Distance3
node 3 via 3 Distance0
node 4 via 4 Distance2

State value for router 4 is
node 1 via 2 Distance4
node 2 via 2 Distance1
```

**Bottom Screenshot:**

```
99 1 2 0

State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance3
node 3 via 3 Distance5
node 4 via 2 Distance4

State value for router 2 is
node 1 via 1 Distance3
node 2 via 2 Distance0
node 3 via 4 Distance3
node 4 via 4 Distance1

State value for router 3 is
node 1 via 1 Distance5
node 2 via 4 Distance3
node 3 via 3 Distance0
node 4 via 4 Distance2

State value for router 4 is
node 1 via 2 Distance4
node 2 via 2 Distance1
node 3 via 3 Distance2
node 4 via 4 Distance0

...Program finished with exit code 0
Press ENTER to exit console.
```

The IDE interface includes a menu bar with options like Run, Debug, Stop, Share, Save, and Beautify. The right sidebar shows panels for Call Stack, Local Variables, Registers, Display Expressions, and Breakpoints and Watchpoints.

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

CODE:

```
class
Graph():

    def __init__(self, vertices):

        self.V = vertices

        self.graph = [[0 for column in range(vertices)]
                       for row in range(vertices)]

    def print_solution(self, dist):

        print("Vertex \tDistance from Source")

        for node in range(self.V):

            print(node, "\t", dist[node])

    def min_distance(self, dist, sptSet):

        min = 9999

        for v in range(self.V):

            if dist[v] < min and sptSet[v] == False:

                min = dist[v]

                min_index = v

        return min_index

    def add_edge(self, src, dest, weight):

        self.graph[src][dest] = self.graph[dest][src] = weight
```

```

def dijkstra(self, src):

    dist = [9999] * self.V
    dist[src] = 0
    sptSet = [False] * self.V

    for cout in range(self.V):
        u = self.min_distance(dist, sptSet)
        sptSet[u] = True
        for v in range(self.V):
            if self.graph[u][v] > 0 and sptSet[v] == False and dist[v] > dist[u] +
self.graph[u][v]:
                dist[v] = dist[u] + self.graph[u][v]

    self.print_solution(dist)

g = Graph(int(input("Enter number of nodes in the topology: ")))
e = int(input("Enter number of edges: "))

for i in range(e):
    src, dest, cost = [int(_) for _ in input("Enter [SRC] [DEST] [WEIGHT]: ").split(' ')]
    g.add_edge(src, dest, cost)

src = int(input("Enter [SRC] to find costs: "))
g.dijkstra(src)

```



## OUTPUT:

The screenshot shows an online C++ compiler interface. The main editor displays a Python 3 code snippet for a graph class. The output console shows the execution of the program, which prompts the user to enter the number of nodes (5) and edges (6), followed by six edges with their source, destination, and weight. The program then prints the shortest distance from source 0 to each vertex (1, 2, 3, 4).

```
1 class Graph():
2     def __init__(self, vertices):
3         self.V = vertices
4         self.graph = [[0 for column in range(vertices)]
5                       for row in range(vertices)]
6
7     def print_solution(self, dist):
8         print("Vertex \tDistance from Source")
9         for node in range(self.V):
10             print(node, dist[node])
```

input

```
Enter number of nodes in the topology: 5
Enter number of edges: 6
Enter [SRC] [DEST] [WEIGHT]: 0 1 2
Enter [SRC] [DEST] [WEIGHT]: 0 2 1
Enter [SRC] [DEST] [WEIGHT]: 0 3 4
Enter [SRC] [DEST] [WEIGHT]: 1 4 3
Enter [SRC] [DEST] [WEIGHT]: 2 4 1
Enter [SRC] [DEST] [WEIGHT]: 3 4 8
Enter [SRC] to find costs: 0
Vertex Distance from Source
0 0
1 2
2 1
3 4
4 2

...Program finished with exit code 0
Press ENTER to exit console.
```

Call Stack

| # | Function | File:Line |
|---|----------|-----------|
|---|----------|-----------|

Local Variables

| Variable | Value |
|----------|-------|
|----------|-------|

Registers

| Register | Value |
|----------|-------|
|----------|-------|

Display Expressions

| Expression/Value |
|------------------|
|------------------|

Breakpoints and Watchpoints

| # | Description |
|---|-------------|
|---|-------------|

25°C Partly cloudy 19:15 09-01-2022

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

CODE:

```
class
LeakyBucket():

    def __init__(self, bucket_size, output_rate, input_packets):

        self.size = bucket_size

        self.orate = output_rate

        self.istream = input_packets


    def congestion_control(self):

        for x in range(len(self.istream)):

            packet_size = self.istream[x]

            print(f"Packet No: {x} Packet Size: {packet_size}")


            if packet_size > self.size:

                print("\t Bucket Overflow")

            else:

                while packet_size > self.orate:

                    print(f"\t {self.orate} bytes sent")

                    packet_size -= self.orate


                if packet_size:

                    print(f"\t Last {packet_size} bytes sent")


            print("\t Bucket output successful \n")


        bucket_size = int(input("Enter Bucket Size: "))

        output_rate = int(input("Enter Output Rate: "))

        input_packets = list(map(int, input("Enter Input Packets: ").split()))
```

```

network = LeakyBucket(bucket_size, output_rate, input_packets)

network.congestion_control()

```

## OUTPUT:

```

ne_c_compiler
input
Enter Bucket Size: 1000
Enter Output Rate: 200
Enter Input Packets: 300 100 200 450 500 550 400
Packet No: 0 Packet Size: 300
    200 bytes sent
    Last 100 bytes sent
    Bucket output successful
Packet No: 1 Packet Size: 100
    Last 100 bytes sent
    Bucket output successful
Packet No: 2 Packet Size: 200
    Last 200 bytes sent
    Bucket output successful
Packet No: 3 Packet Size: 450
    200 bytes sent
    200 bytes sent
    Last 50 bytes sent
    Bucket output successful
Packet No: 4 Packet Size: 500
    200 bytes sent
    200 bytes sent
    Last 100 bytes sent
    Bucket output successful
Packet No: 5 Packet Size: 550
    200 bytes sent
    200 bytes sent
    Last 150 bytes sent
    Bucket output successful
Packet No: 6 Packet Size: 400
    200 bytes sent
    Last 200 bytes sent
    Bucket output successful

```

```

y/online_c_compiler
input
Enter Bucket Size: 1000
Enter Output Rate: 200
Enter Input Packets: 300 100 200 450 500 550 400
Packet No: 0 Packet Size: 300
    200 bytes sent
    Last 100 bytes sent
    Bucket output successful
Packet No: 1 Packet Size: 100
    Last 100 bytes sent
    Bucket output successful
Packet No: 2 Packet Size: 200
    Last 200 bytes sent
    Bucket output successful
Packet No: 3 Packet Size: 450
    200 bytes sent
    200 bytes sent
    Last 50 bytes sent
    Bucket output successful
Packet No: 4 Packet Size: 500
    200 bytes sent
    200 bytes sent
    Last 100 bytes sent
    Bucket output successful
Packet No: 5 Packet Size: 550
    200 bytes sent
    200 bytes sent
    Last 150 bytes sent
    Bucket output successful
Packet No: 6 Packet Size: 400
    200 bytes sent
    Last 200 bytes sent
    Bucket output successful
... Program finished with exit code 0
Press ENTER to exit console.

```

5.

**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 to the requested file if present.**

CODE:

**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()
```

**Client:**

```
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())

print(f"Recieved from {serverName}: ")
```

```

filecontents = clientSocket.recv(1024).decode()

print('\nFrom Server:\n')

print(filecontents)

clientSocket.close()

```

## OUTPUT:

|   |  |
|---|--|
| <pre> Administrator: Command Prompt - python server.py  Microsoft Windows [Version 10.0.19042.1415] (c) Microsoft Corporation. All rights reserved.  C:\WINDOWS\system32&gt;cd C:\Users\Saquib\Desktop\TCP  C:\Users\Saquib\Desktop\TCP&gt;python server.py ----- Server ----- connected by: ('127.0.0.1', 53933) received Filename: test sent: b'File test not found'  received Filename: C:\Users\Saquib\Desktop\TCP\test sent: b'File C:\\Users\\Saquib\\Desktop\\TCP\\test not found'  received Filename: C:\Users\Saquib\Desktop\TCP\test.txt sent: b'this is a test.\nWelcome to Sockets!' </pre> | <pre> Administrator: Command Prompt - python client.py  Microsoft Windows [Version 10.0.19042.1415] (c) Microsoft Corporation. All rights reserved.  C:\WINDOWS\system32&gt;cd C:\Users\Saquib\Desktop\TCP  C:\Users\Saquib\Desktop\TCP&gt;python client.py ----- Client ----- Enter file name: test Sent: test Received: File test not found  Enter file name: C:\Users\Saquib\Desktop\TCP\test Sent: C:\Users\Saquib\Desktop\TCP\test Received: File C:\Users\Saquib\Desktop\TCP\test not found  Enter file name: C:\Users\Saquib\Desktop\TCP\test.txt Sent: C:\Users\Saquib\Desktop\TCP\test.txt Received: this is a test. Welcome to Sockets!  Enter file name: </pre> |
|---|--|

**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:-**

**Server:**

```
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()
```

**Client:**

```
from socket import *

serverName = "127.0.0.1"

serverPort = 12000

clientSocket = socket(AF_INET, SOCK_DGRAM)
```

```

sentence = input("\nEnter file name: ")
clientSocket.sendto(sentence.encode(), (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:

The image shows two side-by-side screenshots of Windows Command Prompts, both titled "Administrator: Command Prompt - python server.py".

The left screenshot shows the server's execution:

```

Microsoft Windows [Version 10.0.19042.1415]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\Saquib\Desktop\UDP

C:\Users\Saquib\Desktop\UDP>python client
python: can't open file 'C:\Users\Saquib\Desktop\UDP\client': [Errno 2] No such file or directory

C:\Users\Saquib\Desktop\UDP>python server.py
----- Server -----
Received Filename: C:\Users\Saquib\Desktop\UDP\test.txt From: ('127.0.0.1', 63050)
Sent: b'this is a test.\nWelcome to Sockets!' To: ('127.0.0.1', 63050)

```

The right screenshot shows the client's execution:

```

Microsoft Windows [Version 10.0.19042.1415]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\Saquib\Desktop\UDP

C:\Users\Saquib\Desktop\UDP>python client.py
----- Client -----
Enter file to request from server: C:\Users\Saquib\Desktop\UDP\test.txt
Sent: C:\Users\Saquib\Desktop\UDP\test.txt
Received: this is a test.
Welcome to Sockets!

Enter file to request from server:

```