**Task-1**

**Study of basic network commands and network configuration commands.**

ping, tracert, ipconfig, hostname, nslookup, netstat

**Task-2**

**Construct Detecting error using CRC-CCITT.**

while True:

print("\n1. CRC-12\n2. CRC-16\n3. CRC-CCITT\n4. Exit\n")

option = int(input("Enter your option: "))

if option == 4:

break

elif option in [1, 2, 3]:

g = ["1100000001111", "11000000000000101", "10001000000100001"][option - 1]

t = input("\nEnter data: ")

a = len(t)

N = len(g)

t += '0' \* (N - 1)

# CRC calculation

cs = t

for \_ in range(len(t)):

cs = ''.join(['0' if cs[i] == g[i] else '1' for i in range(1, len(cs))]) + t[\_]

if cs[0] == '1':

cs = cs[1:] + '0'

print("\nChecksum is:", cs)

print("\nGenerating polynomial:", g)

t = t[:a] + cs[a:]

print("\nModified data is:", t)

# Test error detection

if int(input("\nTest error detection? (0 for yes, 1 for no): ")) == 0:

e = int(input("\nEnter the position where error is to be inserted: "))

if 0 < e <= a + N - 1:

t = t[:e-1] + ('1' if t[e-1] == '0' else '0') + t[e:]

print("\nErroneous data:", t)

# CRC calculation after error insertion

cs = t

for \_ in range(len(t)):

cs = ''.join(['0' if cs[i] == g[i] else '1' for i in range(1, len(cs))]) + t[\_]

if cs[0] == '1':

cs = cs[1:] + '0'

print("\nError detected\n" if '1' in cs else "\nNo error detected\n")

else:

print("Invalid option")

**Task-3**

**3.Implementation of Bit Stuffing**

**1**

ip = input("Enter input bit sequence: ")

pre\_post = '01111110'

op = [pre\_post] + [c if ip[i-1:i+4] != '11111' else '0' for i, c in enumerate(ip)]

print("Output\n------\nStuffed Bit Sequence is:", ''.join(op))

decode\_op = [c for i, c in enumerate(op) if i > 7 and op[i-1:i+1] != '11']

print("Destuffed Bit Sequence is:", ''.join(decode\_op))

**2**

ip = input("Enter input bit sequence: ")

pre\_post = '01111110'

op = [pre\_post] + [c if ip[i-1:i+4] != '11111' else '0' for i, c in enumerate(ip, start=1)]

print("Output\n------\nStuffed Bit Sequence is:", ''.join(op))

decode\_op = ''

i = 8

while i < len(op) - 8:

decode\_op += op[i]

if op[i] == '1':

i += 1

if op[i] == '0':

i += 1

i += 1

print("Destuffed Bit Sequence is:", decode\_op)

**Task-4**

**Implementation of Character Stuffing**

source = input("Enter plain text: ")

# Character stuffing

char\_stuff = ''.join(['dle' + c if source[i:i+3] == 'dle' else c for i, c in enumerate(source)]) + 'dlesctx'

print("After character stuffing:", char\_stuff)

# Character destuffing

char\_destuff = ''.join([source[i] for i in range(len(source)) if source[i:i+3] != 'dle'])

print("After character de-stuffing:", char\_destuff)

**Task-5**

**Implementation of stop and wait protocol**

framesize, sent = int(input("Enter number of frames:\n")), 0

while True:

for i in range(framesize):

print(f"Frame {sent} has been transmitted.")

sent += 1

if sent == framesize: break

if (sent := int(input("\nPlease enter the last acknowledgment received:\n"))) >= framesize: break

**Task-6**

**Implementation of Dijkstra’s algorithm**

M, I = 50, 1000

n = int(input("enter Number of nodes:"))

N = [chr(65+i) for i in range(n)]

D = [[int(input()) for \_ in range(n)] for \_ in range(n)]

S, E = input("Sorcce:"), input("destination:")

s, e = N.index(S), N.index(E)

S = [{'p':-1,'l':I,'v':0} for \_ in range(n)]

S[s].update({'l':0,'v':1})

k = s

while k != e:

for i in [i for i in range(n) if D[k][i] and not S[i]['v']]:

l = S[k]['l'] + D[k][i]

if l < S[i]['l']: S[i].update({'p':k,'l':l})

k = min([i for i in range(n) if not S[i]['v']], key=lambda x: S[x]['l'])

S[k]['v'] = 1

i, k, p = 0, e, [0] \* M

while k >= 0: p[i], k, i = k, S[k]['p'], i + 1

print("\nshortest distance:", S[e]['l'], "\nshortest path: ", end="")

print("->".join([N[p[i]] for i in range(i - 1, -1, -1)]))

**Task-7**

**Implementation Distance vector algorithm**

n = int(input("Enter the number of nodes in the graph: "))

edge = [[int(input(f"\nIs there any edge from {i+1} to {j+1}? ")) for j in range(n)] for i in range(n)]

x, y = map(int, input("\nEnter the source and destination nodes: ").split())

cost = [0] \* n

for i in range(n):

if edge[i][x-1] or edge[x-1][i]:

cost[i] = int(input(f"\nEnter the cost of node {x} to its neighbour {i+1}: ")) + int(input(f"Enter the Routing table entry from {i+1} to {y}: "))

delay, d = min((cost[i], i+1) for i in range(n) if cost[i])

print(f"\nEstimated cost from node {x} to {y} is {delay} via the node {d}.")

**Task-8**

**. Implementation of Congestion control using leaky bucket algorithms**

import time

import random

bucket\_size = 512

op = int(input("Enter output rate: "))

for i in range(1, 6):

time.sleep(2)

pkt\_size = random.randint(1, 1000)

print(f"\nPacket no: {i} \tPacket size: {pkt\_size}")

if pkt\_size > bucket\_size:

print("\n\t\tBucket overflow")

else:

time.sleep(2)

while pkt\_size > op:

print(f"\n\t\t{op} bytes outputted.")

pkt\_size -= op

time.sleep(2)

print(f"\n\t\tLast {pkt\_size} bytes sent\t") if pkt\_size > 0 else None

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

**Task-9**

**Implementation using Socket TCP both client and server**

**server**

import socket

MAX = 80

PORT = 8080

with socket.socket() as sockfd:

sockfd.bind(('localhost', PORT))

sockfd.listen(5)

print("Server listening..")

conn, addr = sockfd.accept()

print("Server accepted the client...")

while True:

buff = conn.recv(MAX).decode()

print("From client:", buff, "\tTo client: ", end="")

msg = input()

conn.send(msg.encode())

if msg.lower() == "exit":

print("Server Exit...")

break

conn.close()

**client**

import socket

MAX = 80

PORT = 8080

sockfd = socket.socket()

sockfd.connect(('127.0.0.1', PORT))

print("Connected to the server..")

while True:

sockfd.send(input("Enter the string: ").encode())

buff = sockfd.recv(MAX).decode()

print("From Server:", buff)

if buff.strip().lower() == "exit":

print("Client Exit...")

break

sockfd.close()

**Java**

**Client**

import java.io.\*;

import java.net.\*;

public class MyClient {

public static void main(String[] args) {

try{

Socket s=new Socket("localhost",6666);

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

dout.writeUTF("Hello Server");

dout.flush();

dout.close();

s.close();

}catch(Exception e){System.out.println(e);}

}

}

**Server**

import java.io.\*;

import java.net.\*;

public class MyServer {

public static void main(String[] args){

try{

ServerSocket ss=new ServerSocket(6666);

Socket s=ss.accept();//establishes connection

DataInputStream dis=new DataInputStream(s.getInputStream());

String str=(String)dis.readUTF();

System.out.println("message= "+str);

ss.close();

}catch(Exception e)

{

System.out.println(e);

}

}

}

**Task-10**

**Implementation using Socket UDP both client and server**

**Server**

import socket

PORT = 8080

MAXLINE = 1024

hello = "Hello from server"

sockfd = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sockfd.bind(('127.0.0.1', PORT))

while True:

buffer, cliaddr = sockfd.recvfrom(MAXLINE)

print("Client:", buffer.decode())

sockfd.sendto(hello.encode(), cliaddr)

print("Hello message sent.")

**Client**

import socket

PORT = 8080

MAXLINE = 1024

hello = "Hello from client"

sockfd = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

sockfd.sendto(hello.encode(), ('127.0.0.1', PORT))

print("Hello message sent.")

buffer, \_ = sockfd.recvfrom(MAXLINE)

print("Server:", buffer.decode())

sockfd.close()

**Java**

**Server**

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.SocketException;

public class server

{

public static void main(String[] args) throws IOException{

DatagramSocket ds = new DatagramSocket(1234);

byte[] receive = new byte[65535];

DatagramPacket DpReceive = null;

while (true){

DpReceive = new DatagramPacket(receive, receive.length);

ds.receive(DpReceive);

System.out.println("Client:-" + data(receive));

if (data(receive).toString().equals("bye")){

System.out.println("Client sent bye.....EXITING");

break;

}

receive = new byte[65535];

}

}

// A utility method to convert the byte array

// data into a string representation.

public static StringBuilder data(byte[] a){

if (a == null)

return null;

StringBuilder ret = new StringBuilder();

int i = 0;

while (a[i] != 0){

ret.append((char) a[i]);

i++;

}

return ret;

}

}

**Client**

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class client{

public static void main(String[] args) throws IOException{

Scanner sc = new Scanner(System.in);

DatagramSocket ds = new DatagramSocket();

InetAddress ip = InetAddress.getLocalHost();

byte buf[] = null;

while (true){

String inp = sc.nextLine();

buf = inp.getBytes();

DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip, 1234);

ds.send(DpSend);

if (inp.equals("bye"))

break;

}

}

}

EXP-