# Name:Kanala Bhuvana Reddy

# USN: 1BM19CS069

CN-LAB PROGRAMS 6-10

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

def xor1(a, b):

x = ""

# print(len(a),len(b)) for i in range(1, len(a)):

if a[i] == b[i]:

x += "0"

else: x += "1"

return x

def modulo2(divident, divisor): divlen = len(divisor) temp = divident[0:divlen]

# print(temp) while(divlen < len(divident)):

if temp[0] == "1":

temp = xor1(temp, divisor)+divident[divlen]

else:

temp = temp[1:divlen]+divident[divlen]

# print(temp) divlen += 1 # print(temp) if temp[0] == "1": temp = xor1(temp, divisor)

# return "0"+temp # print(len(temp),) if len(temp) < len(divisor):

return "0"+temp

return temp

def encode(data, key):

append = data+"0"\*(len(key))

# print(code) rem = modulo2(append, key) print("remaindar="+rem) code = data+rem print("code="+code)

rem = modulo2(code, key) print("Remaindar we get when we do not have error="+rem) code = code.replace("011", "101") rem = modulo2(code, key)

print("Remaindar we get when we have error="+rem)

def polytobin(string):

keys = [] key = "" for i in string:

if i == '+': keys.append(int(key[1:])) key = "" continue

key += i

if key != "": keys.append(0)

bina = "" j = 0 print(keys) for i in range(keys[0], -1, -1):

if i == (keys[j]): bina += "1" j += 1

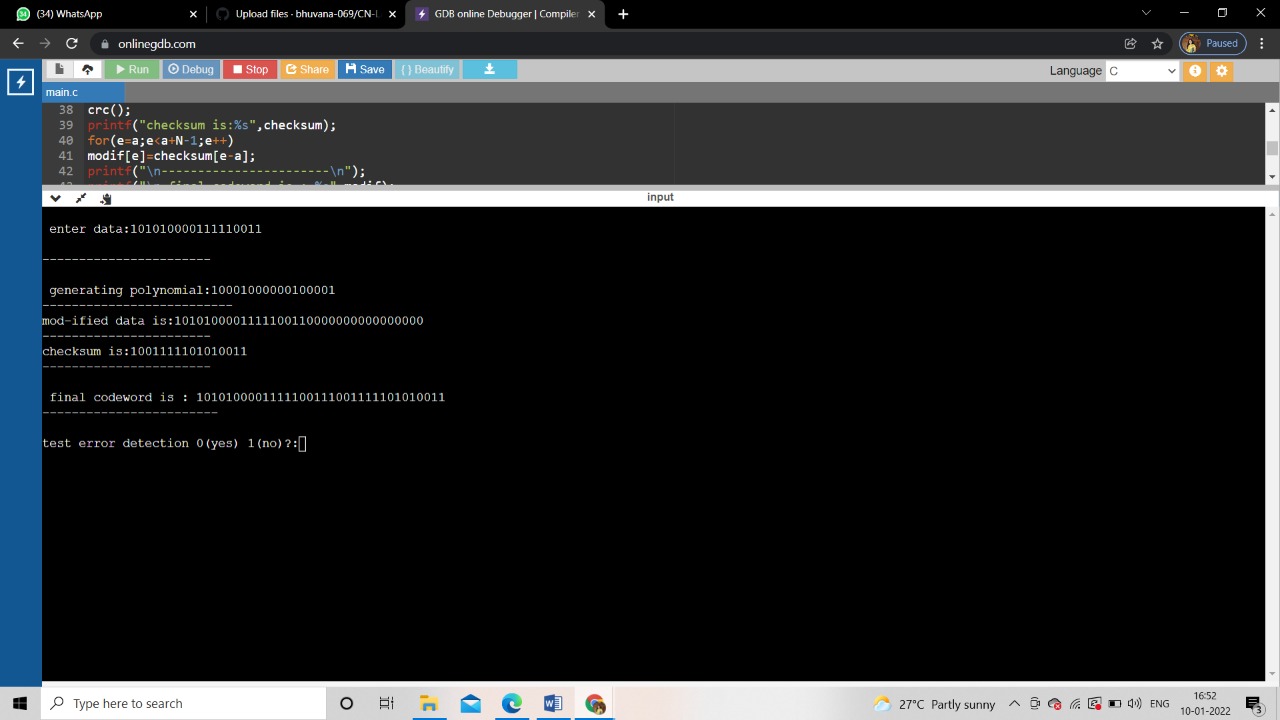
else:

bina += "0"

print(bina) return bina

string = input("Enter the key polynomial:\n") key = polytobin(string) string = input("Enter the data polynomial:\n") data = polytobin(string) print(key, data) encode(data, key)

OUTPUT:



7. Write a program for the distance vector algorithm to find a suitable path for transmission. class Graph:

def \_\_init\_\_(self, vertices):

self.V = vertices self.graph = []

def add\_edge(self, s, d, w): self.graph.append([s, d, w])

def print\_solution(self, dist, src, next\_hop):

print("Routing table for ", src) print("Dest \t Cost \t Next Hop") for i in range(self.V): print("{0} \t {1} \t {2}".format(i, dist[i], next\_hop[i])) def bellman\_ford(self, src):

dist = [99] \* self.V dist[src] = 0 next\_hop = {src: src} for \_ in range(self.V - 1):

for s, d, w in self.graph:

if dist[s] != 99 and dist[s] + w < dist[d]:

dist[d] = dist[s] + w if s == src:

next\_hop[d] = d elif s in next\_hop:

next\_hop[d] = next\_hop[s]

for s, d, w in self.graph:

if dist[s] != 99 and dist[s] + w < dist[d]:

print("Graph contains negative weight cycle") return

self.print\_solution(dist, src, next\_hop) def main(): matrix = [] print("Enter the no. of routers:") n = int(input())

print("Enter the adjacency matrix : Enter 99 for infinity") for i in range(0, n): a = list(map(int, input().split(" "))) matrix.append(a)

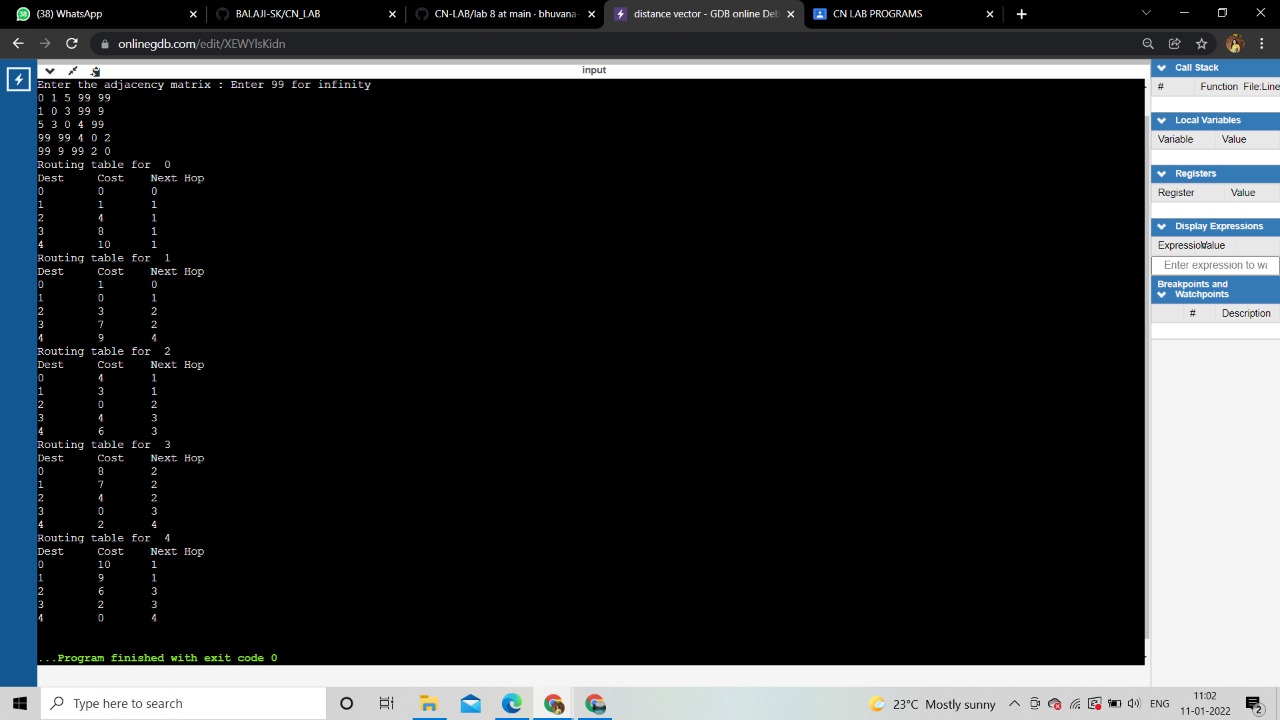
g = Graph(n) for i in range(0, n): for j in range(0, n):

g.add\_edge(i, j, matrix[i][j])

for k in range(0, n):

g.bellman\_ford(k)

main()

OUTPUT: 

8.Implement Dijkstra’s algorithm to compute the shortest path for a given topology.

#include<bits/stdc++.h> using namespace std; #define V 5

int minDistance(int dist[], bool sptSet[])

{

int min = 9999, 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 printPath(int parent[], int j)

{ if (parent[j] == - 1) return; printPath(parent, parent[j]);

cout<<j<<" ";

}

void printSolution(int dist[], int n, int parent[])

{ int src = 0; cout<<"Vertex\t Distance\tPath"<<endl; for (int i = 1; i < V; i++)

{ cout<<"\n"<<src<<" -> "<<i<<" \t "<<dist[i]<<"\t\t"<<src<<" "; printPath(parent, i);

} }

void dijkstra(int graph[V][V], int src)

{ int dist[V]; bool sptSet[V]; int parent[V];

for (int i = 0; i < V; i++)

{ parent[0] = -1; dist[i] = 9999; 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] + graph[u][v] < dist[v])

{ parent[v] = u; dist[v] = dist[u] + graph[u][v];

} }

printSolution(dist, V, parent);

}

int main()

{ int graph[V][V]; cout<<"Enter the graph (Enter 99 for infinity): "<<endl; for(int i = 0; i<V; i++)

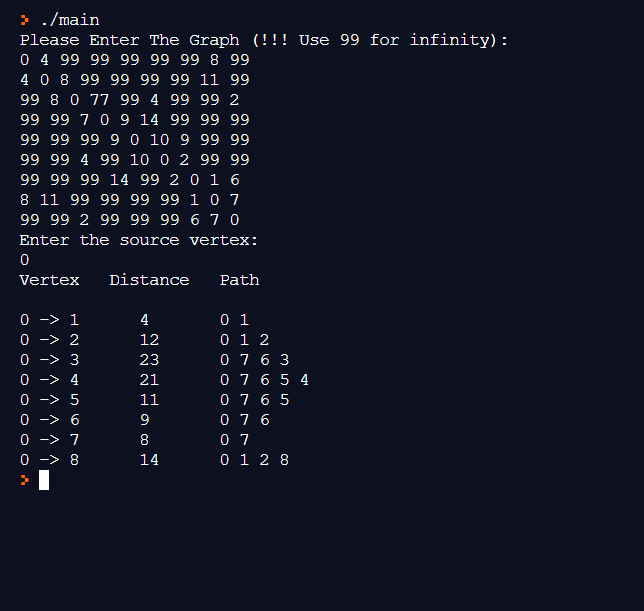
{ for(int j = 0; j<V; j++) cin>>graph[i][j];

} cout<<"Enter the source: "<<endl; int src; cin>>src;

dijkstra(graph, src); cout<<endl; return 0;

}

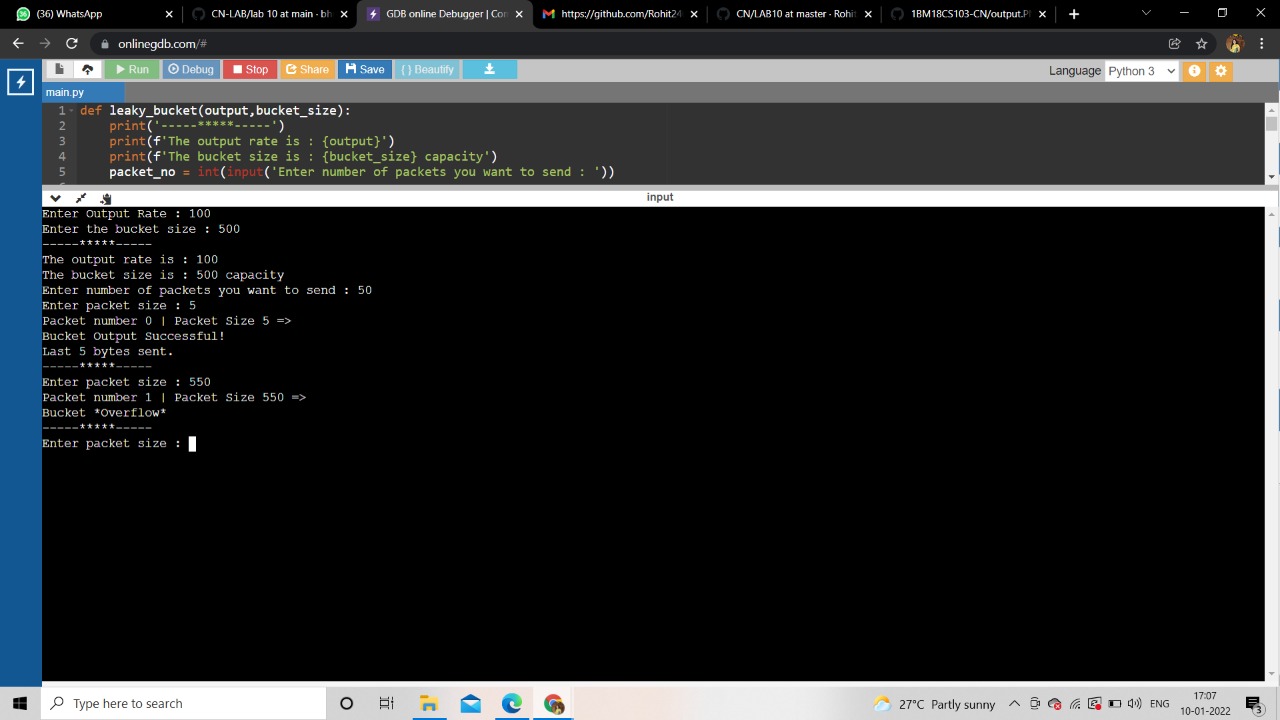
Output:



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

|  |
| --- |
| #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\tBucket overflow"; |
|  | else{ |
|  | sleep(5); |
|  | while(a > b){ |
|  | cout<<"\n\t\t"<<b<<" bytes outputted."; |
|  | a-=b; |
|  | sleep(5); |
|  | } |
|  | if(a > 0) |
|  | cout<<"\n\t\tLast "<<a<<" bytes sent\t"; |
|  | cout<<"\n\t\tBucket output successful"; |
|  | } |
|  | } |
|  | 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; |
|  | } |
|  |  |
|  | /\* |
|  |  |
|  | Enter output rate : 100 |
|  |  |
|  | Packet no 1 Packet size = 267 |
|  | 100 bytes outputted. |
|  | 100 bytes outputted. |
|  | Last 67 bytes sent |
|  | Bucket output successful |
|  | Packet no 2 Packet size = 600 |
|  | Bucket overflow |
|  | Packet no 3 Packet size = 324 |
|  | 100 bytes outputted. |
|  | 100 bytes outputted. |
|  | 100 bytes outputted. |
|  | Last 24 bytes sent |
|  | Bucket output successful |
|  | Packet no 4 Packet size = 658 |
|  | Bucket overflow |
|  | Packet no 5 Packet size = 664 |
|  | Bucket overflow |

Output:



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

#Client.py from socket import \* serverName = '127.0.0.1' serverPort = 12000 clientSocket = socket(AF\_INET, SOCK\_STREAM) clientSocket.connect((serverName,serverPort)) sentence = input("Enter file name: \t ") clientSocket.send(sentence.encode()) filecontents = clientSocket.recv(1024).decode() print ('From Server:', filecontents) clientSocket.close()

#Server.py from socket import \* serverName='127.0.0.1' serverPort = 12000 serverSocket = socket(AF\_INET,SOCK\_STREAM) serverSocket.bind((serverName,serverPort)) serverSocket.listen(1) print ("The server is ready to receive") while 1:

connectionSocket, addr = serverSocket.accept() sentence = connectionSocket.recv(1024).decode() file=open(sentence,"r") l=file.read(1024) connectionSocket.send(l.encode()) file.close() connectionSocket.close()

Output:

12. **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.**

from socket import \* serverName = "127.0.0.1" serverPort = 12000 clientSocket = socket(AF\_INET, SOCK\_DGRAM) sentence = input("Enter file name: ") clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort)) filecontents,serverAddress = clientSocket.recvfrom(2048) print ('From Server:', filecontents) clientSocket.close()

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) file=open(sentence,"r") l=file.read(2048)

serverSocket.sendto(bytes(l,"utf-8"),clientAddress) print("sent back to client",l) file.close()

output:

