# CYCLE 2

# COMPUTER NETWORK LAB

**SAMARTH C SHETTY**
**1BM19CS141**

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

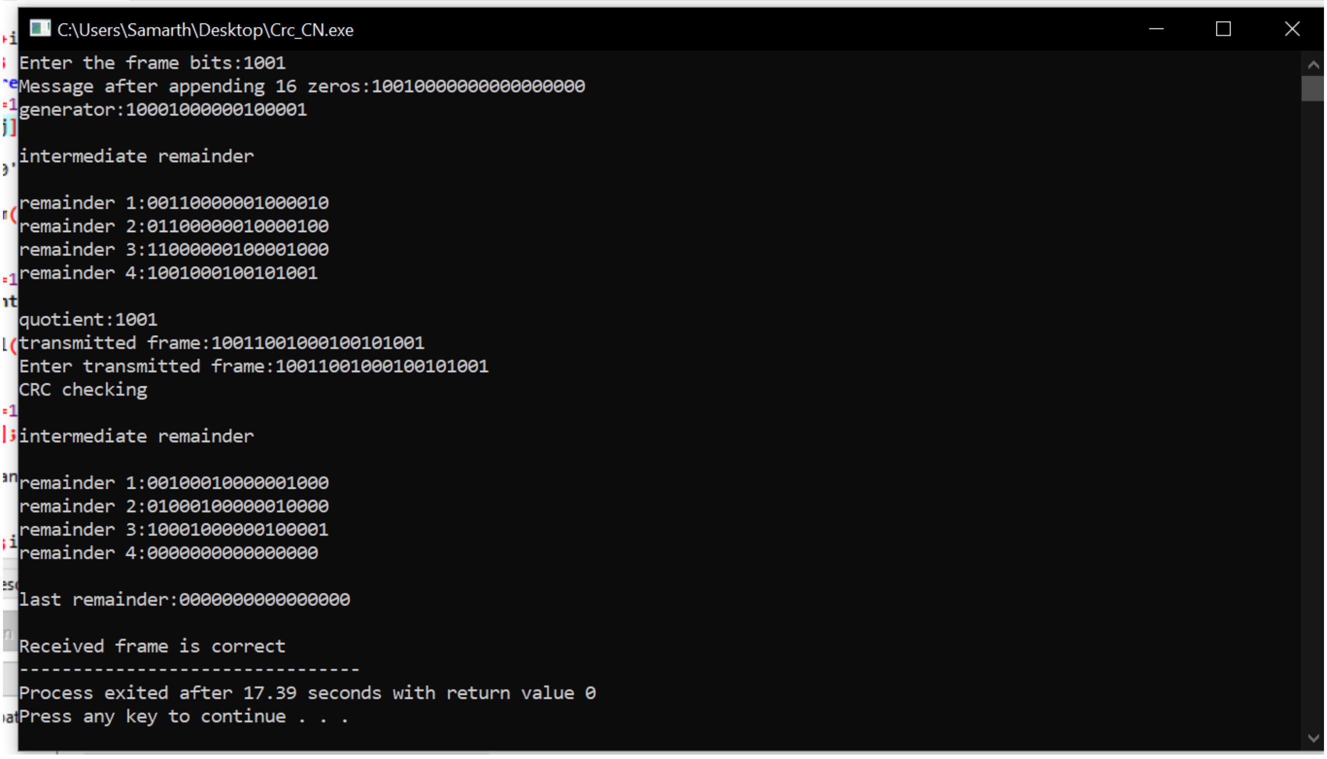
## Program:

```
#include<stdio.h>
char m[50],g[50],r[50],q[50],temp[50];
void caltrans(int);
void crc(int);
void calram();
void shiftl();
int main()
{
int n,i=0;
char ch,flag=0;
printf("Enter the frame bits:");
while((ch=getc(stdin))!='\n')
m[i++]=ch;
n=i;
for(i=0;i<16;i++)
m[n++]='0';
m[n]='\0';
printf("Message after appending 16 zeros:%s",m);
for(i=0;i<=16;i++)
g[i]='0';
g[0]=g[4]=g[11]=g[16]='1';g[17]='\0';
printf("\ngenerator:%s\n",g);
crc(n);
printf("\n\nquotient:%s",q);
```

```
caltrans(n);
printf("\ntransmitted frame:%s",m);
printf("\nEnter transmitted freme:");
scanf("\n%s",m);
printf("CRC checking\n");
crc(n);
printf("\n\nlast remainder:%s",r);
for(i=0;i<16;i++)
if(r[i]!='0')
flag=1;
else
continue;
if(flag==1)
printf("Error during transmission");
else
printf("\n\nReceived freme is correct");
}
void crc(int n)
{
int i,j;
for(i=0;i<n;i++)
temp[i]=m[i];
for(i=0;i<16;i++)
r[i]=m[i];
printf("\nintermediate remainder\n");
for(i=0;i<n-16;i++)
{
if(r[0]=='1')
{
q[i]='1';
calram();
}
else
{
q[i]='0';
shiftl();
}
r[16]=m[17+i];
r[17]='\0';
printf("\nremainder %d:%s",i+1,r);
for(j=0;j<=17;j++)
temp[j]=r[j];
}
q[n-16]='\0';
}
void calram()
{
int i,j;
for(i=1;i<=16;i++)
r[i-1]=((int)temp[i]-48)^((int)g[i]-48)+48;
}
void shiftl()
{
int i;
for(i=1;i<=16;i++)
r[i-1]=r[i];
}
void caltrans(int n)
{
int i,k=0;
for(i=n-16;i<n;i++)
m[i]=((int)m[i]-48)^((int)r[k++]-48)+48;
m[i]='\0';
```

```
}
```

## Output :



```
Enter the frame bits:1001
Message after appending 16 zeros:10010000000000000000
generator:10001000000100001

intermediate remainder

remainder 1:00110000001000010
remainder 2:01100000010000100
remainder 3:11000000100001000
remainder 4:1001000100101001

quotient:1001
transmitted frame:10011001000100101001
Enter transmitted frame:10011001000100101001
CRC checking

intermediate remainder

remainder 1:00100010000001000
remainder 2:01000100000010000
remainder 3:10001000000100001
remainder 4:0000000000000000

last remainder:0000000000000000

Received frame is correct
--------------------------------
Process exited after 17.39 seconds with return value 0
Press any key to continue . . .
```

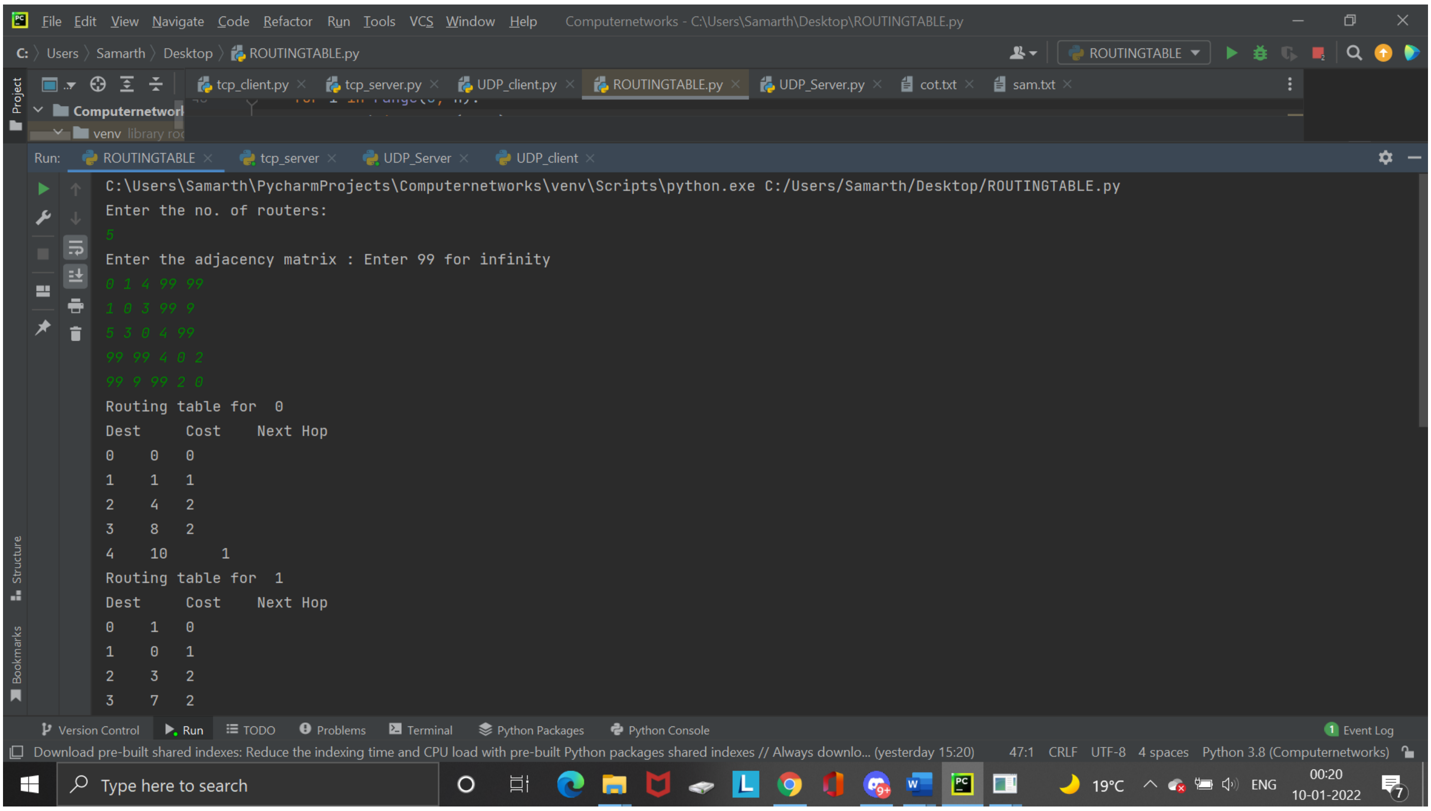
## 2.Write a program for distance vector algorithm to find suitable path for transmission.

## Program

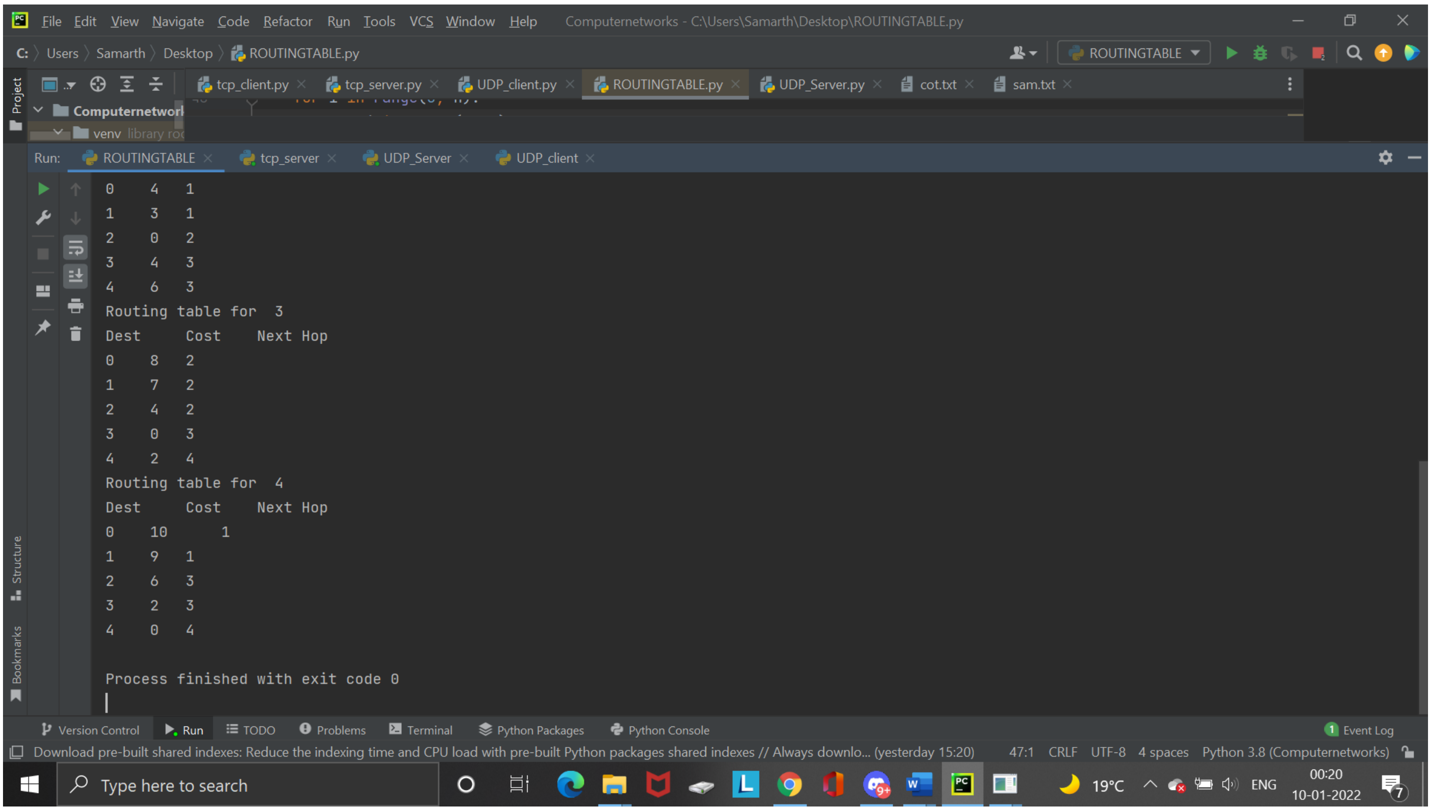
```
class Graph:

   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
inrange(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:



```
C:\Users\Samarth\PycharmProjects\Computernetworks\venv\Scripts\python.exe C:/Users/Samarth/Desktop/ROUTINGTABLE.py
Enter the no. of routers:
5
Enter the adjacency matrix : Enter 99 for infinity
0 1 4 99 99
1 0 3 99 9
5 3 0 4 99
99 99 4 0 2
99 9 99 2 0
Routing table for  0
Dest     Cost    Next Hop
0    0    0
1    1    1
2    4    2
3    8    2
4    10       1
Routing table for  1
Dest     Cost    Next Hop
0    1    0
1    0    1
2    3    2
3    7    2
```



```
0    4    1
1    3    1
2    0    2
3    4    3
4    6    3
Routing table for  3
Dest     Cost    Next Hop
0    8    2
1    7    2
2    4    2
3    0    3
4    2    4
Routing table for  4
Dest     Cost    Next Hop
0    10       1
1    9    1
2    6    3
3    2    3
4    0    4

Process finished with exit code 0
```

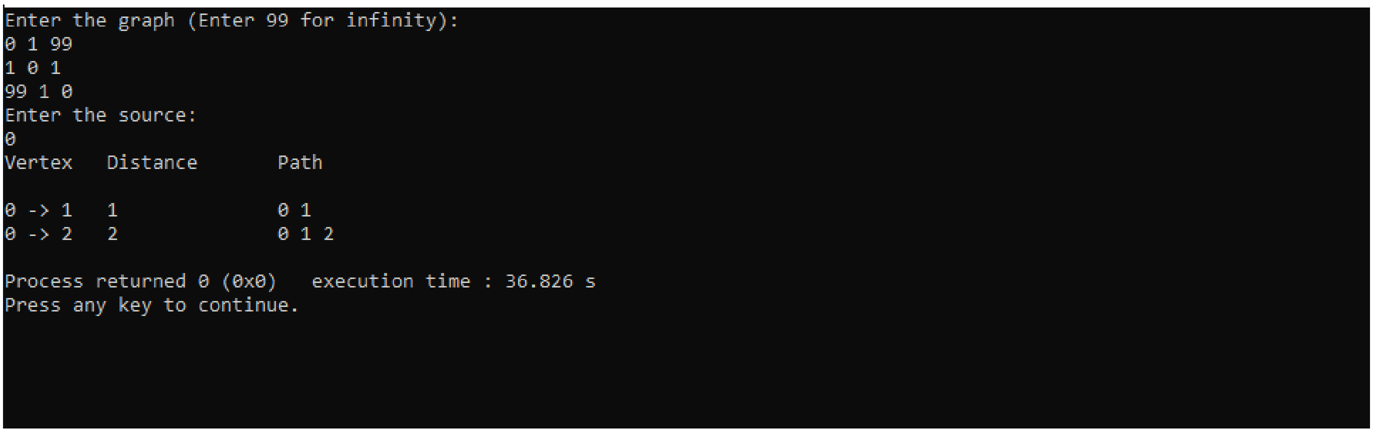
3. Implement Dijkstra’s algorithm to compute the shortest path for a given topology

Program :

```
#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<<"Enterthegraph(Enter99forinfinity):"<<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 :



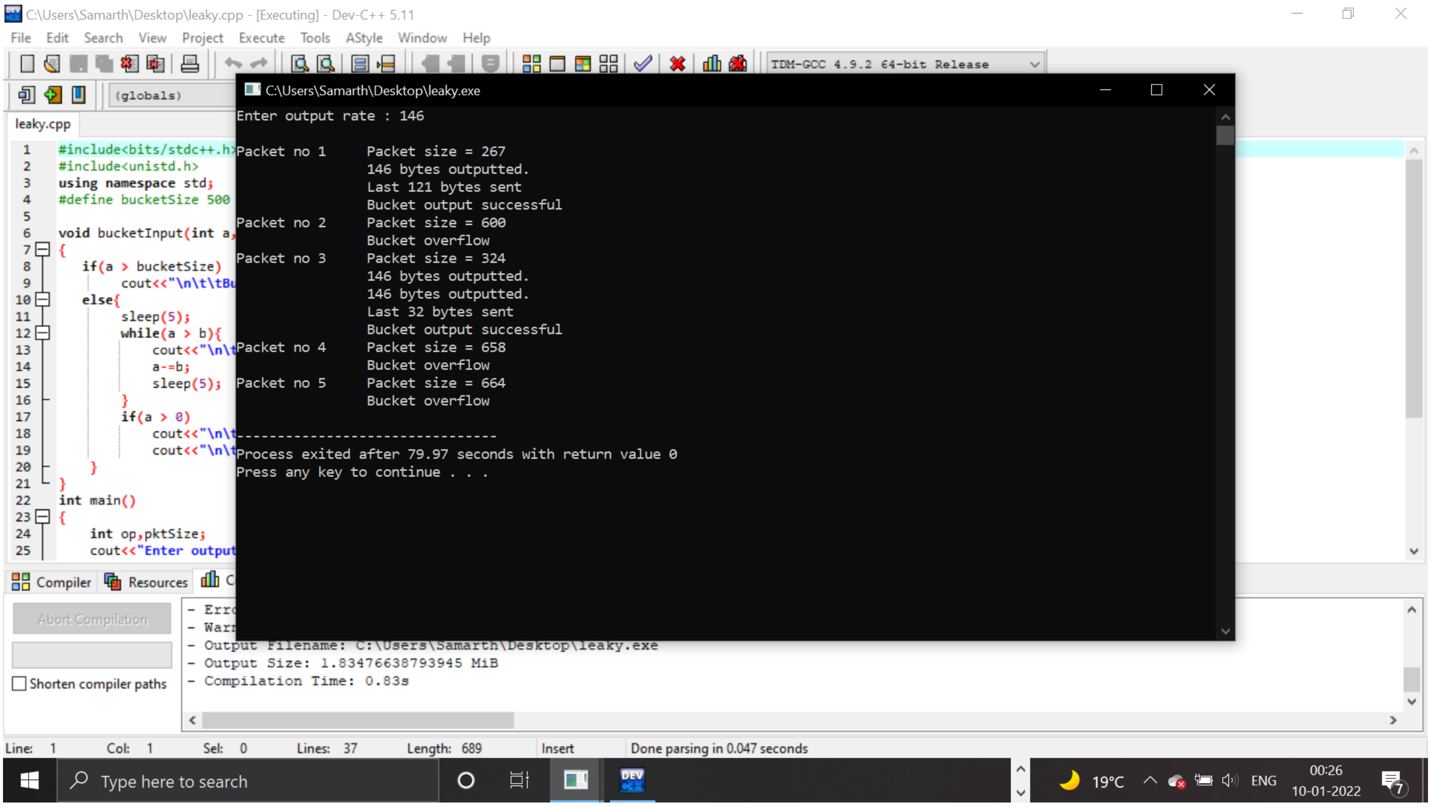
```
Enter the graph (Enter 99 for infinity):
0 1 99
1 0 1
99 1 0
Enter the source:
0
Vertex   Distance       Path

0 -> 1   1              0 1
0 -> 2   2              0 1 2

Process returned 0 (0x0)   execution time : 36.826 s
Press any key to continue.
```

4) **Write a program for congestion control using Leaky bucket algorithm**

**Program :**

```
#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;
}
```

# Output :



C:\Users\Samarth\Desktop\leaky.cpp - [Executing] - Dev-C++ 5.11

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

void bucketInput(int a,
{
    if(a > bucketSize)
        cout<<"\n\t\tBu
    else{
        sleep(5);
        while(a > b){
            cout<<"\n\t
            a-=b;
            sleep(5);
        }
        if(a > 0)
            cout<<"\n\t
            cout<<"\n\t
    }
}
int main()
{
    int op,pktSize;
    cout<<"Enter output
```

C:\Users\Samarth\Desktop\leaky.exe

```
Enter output rate : 146

Packet no 1     Packet size = 267
                146 bytes outputted.
                Last 121 bytes sent
                Bucket output successful
Packet no 2     Packet size = 600
                Bucket overflow
Packet no 3     Packet size = 324
                146 bytes outputted.
                146 bytes outputted.
                Last 32 bytes sent
                Bucket output successful
Packet no 4     Packet size = 658
                Bucket overflow
Packet no 5     Packet size = 664
                Bucket overflow

--------------------------------
Process exited after 79.97 seconds with return value 0
Press any key to continue . . .
```

```
- Erro
- Warn
- Output Filename: C:\Users\Samarth\Desktop\leaky.exe
- Output Size: 1.83476638793945 MiB
- Compilation Time: 0.83s
```

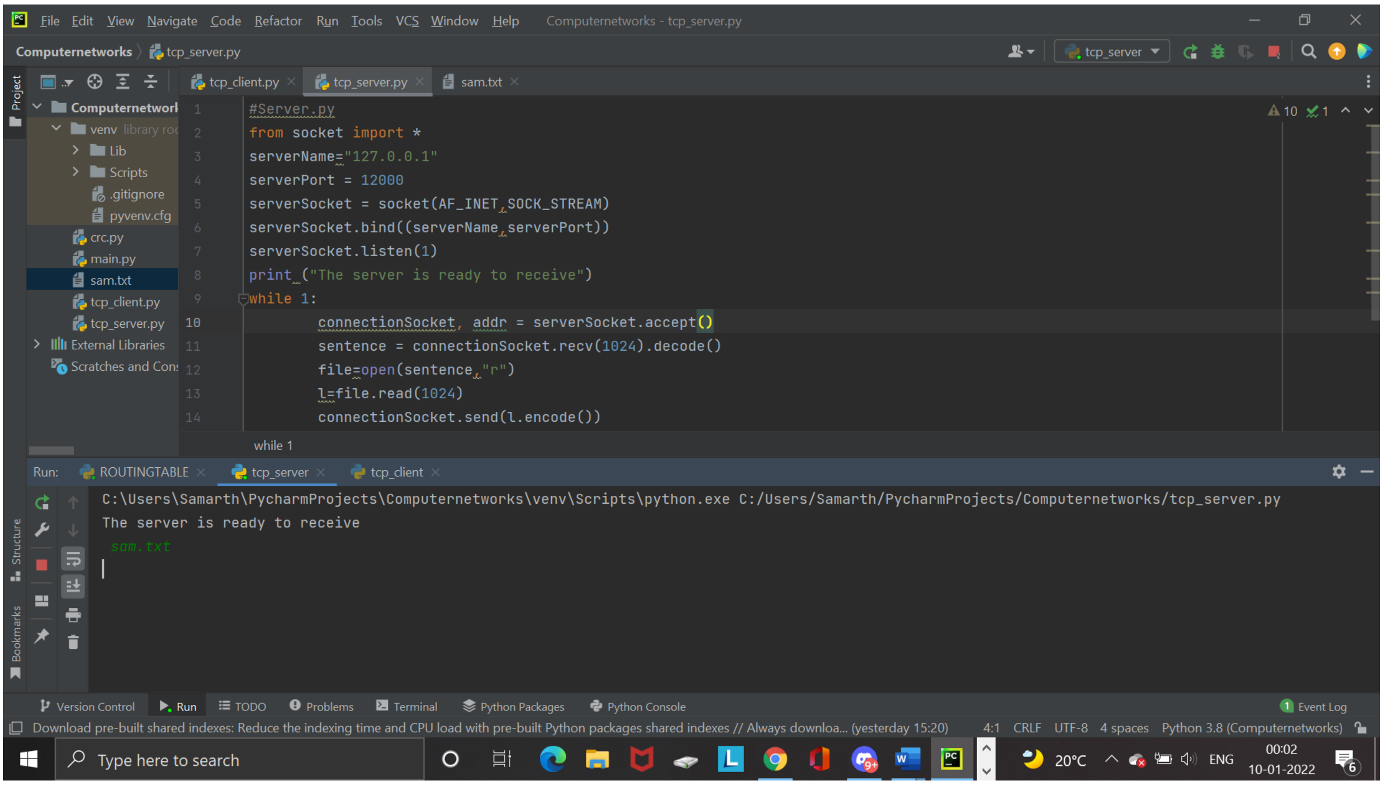
Line: 1    Col: 1    Sel: 0    Lines: 37    Length: 689    Insert    Done parsing in 0.047 seconds

## 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.

## Program :

```
#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")
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()
```

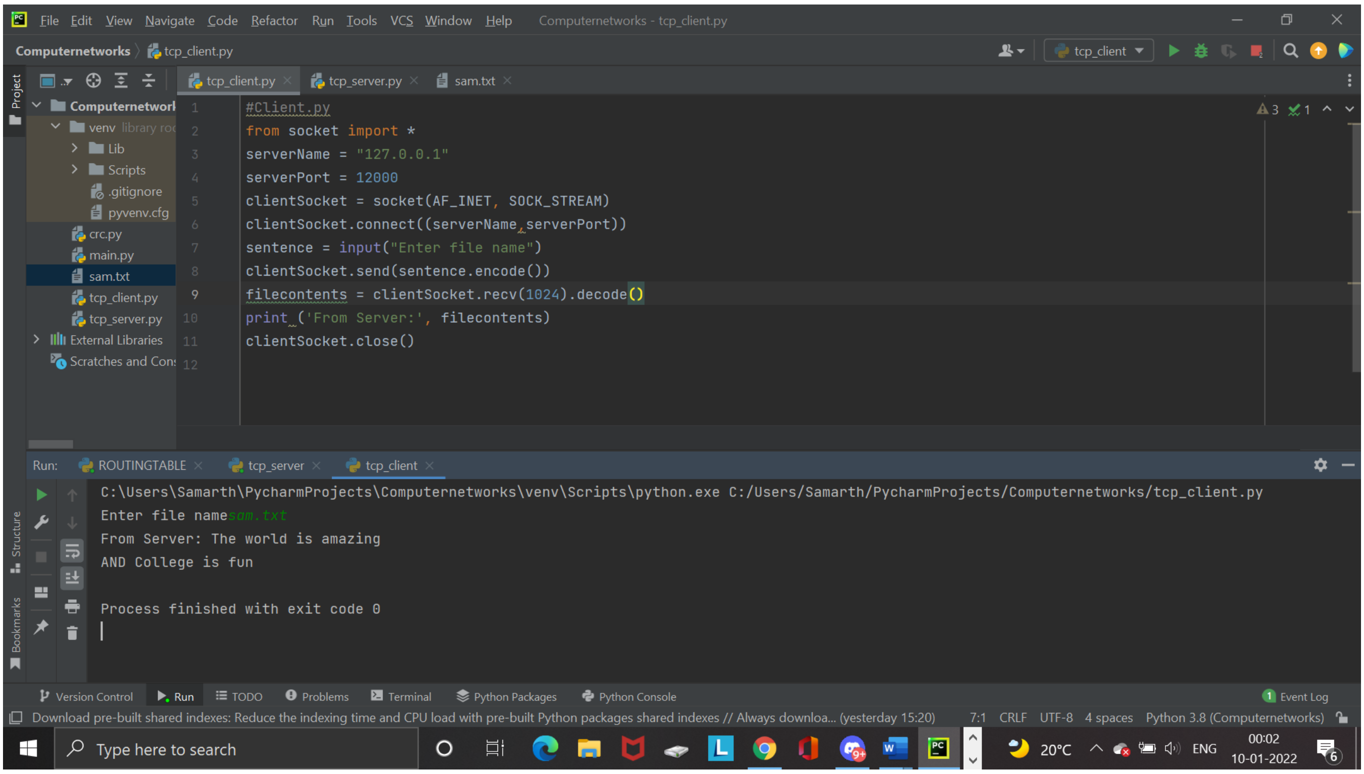
# Output



File Edit View Navigate Code Refactor Run Tools VCS Window Help Computernetworks - tcp_server.py
Computernetworks tcp_server.py
tcp_client.py tcp_server.py sam.txt
Computernetworks
venv library root
Lib
Scripts
.gitignore
pyvenv.cfg
crc.py
main.py
sam.txt
tcp_client.py
tcp_server.py
External Libraries
Scratches and Consoles
while 1
Run: ROUTINGTABLE tcp_server tcp_client
Version Control Run TODO Problems Terminal Python Packages Python Console Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always downloa... (yesterday 15:20) 4:1 CRLF UTF-8 4 spaces Python 3.8 (Computernetworks)
Type here to search
20°C ENG 00:02 10-01-2022

```
#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())
```

```
C:\Users\Samarth\PycharmProjects\Computernetworks\venv\Scripts\python.exe C:/Users/Samarth/PycharmProjects/Computernetworks/tcp_server.py
The server is ready to receive
 sam.txt
```



File Edit View Navigate Code Refactor Run Tools VCS Window Help Computernetworks - tcp_client.py
Computernetworks tcp_client.py
tcp_client.py tcp_server.py sam.txt
Computernetworks
venv library root
Lib
Scripts
.gitignore
pyvenv.cfg
crc.py
main.py
sam.txt
tcp_client.py
tcp_server.py
External Libraries
Scratches and Consoles
Run: ROUTINGTABLE tcp_server tcp_client
Version Control Run TODO Problems Terminal Python Packages Python Console Event Log
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always downloa... (yesterday 15:20) 7:1 CRLF UTF-8 4 spaces Python 3.8 (Computernetworks)
Type here to search
20°C ENG 00:02 10-01-2022

```
#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")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ('From Server:', filecontents)
clientSocket.close()
```

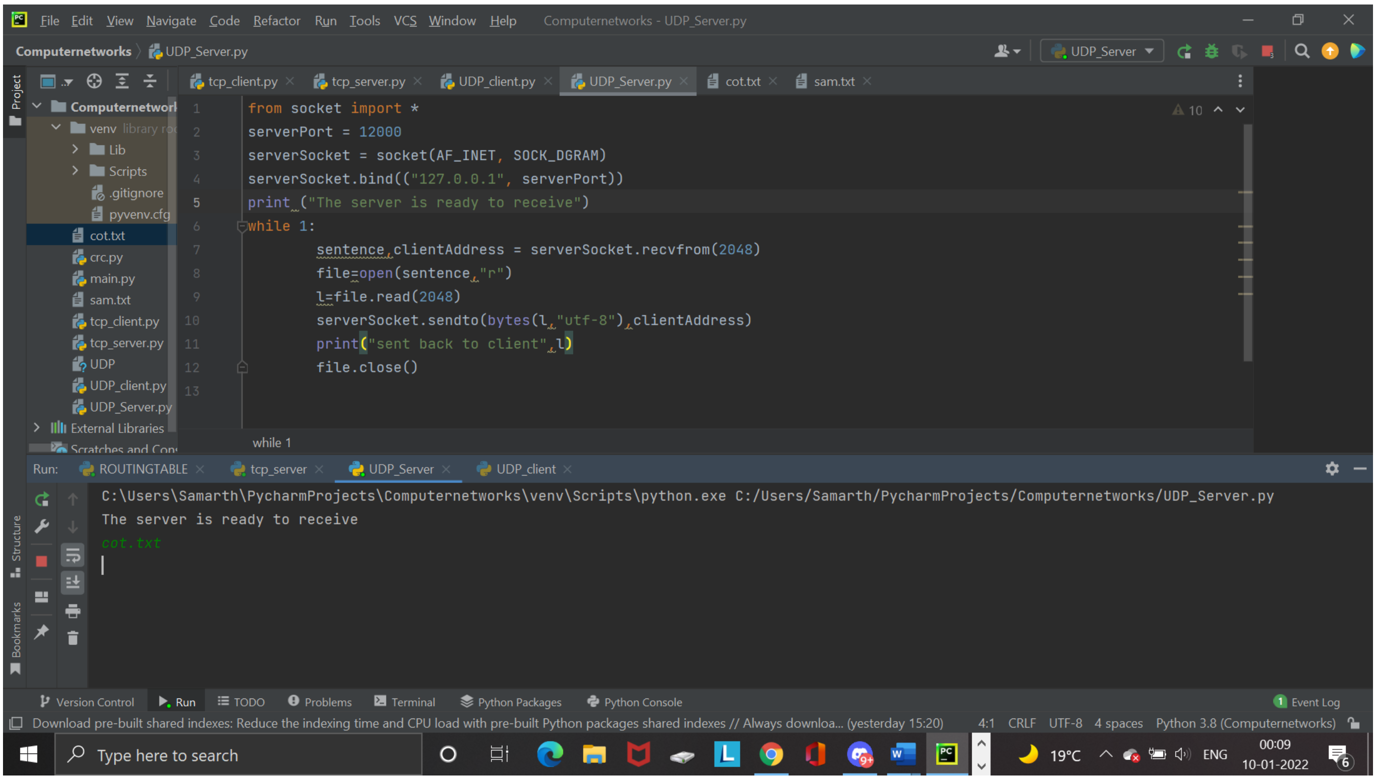
```
C:\Users\Samarth\PycharmProjects\Computernetworks\venv\Scripts\python.exe C:/Users/Samarth/PycharmProjects/Computernetworks/tcp_client.py
Enter file namesam.txt
From Server: The world is amazing
AND College is fun

Process finished with exit code 0
```

## 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.

## Program :

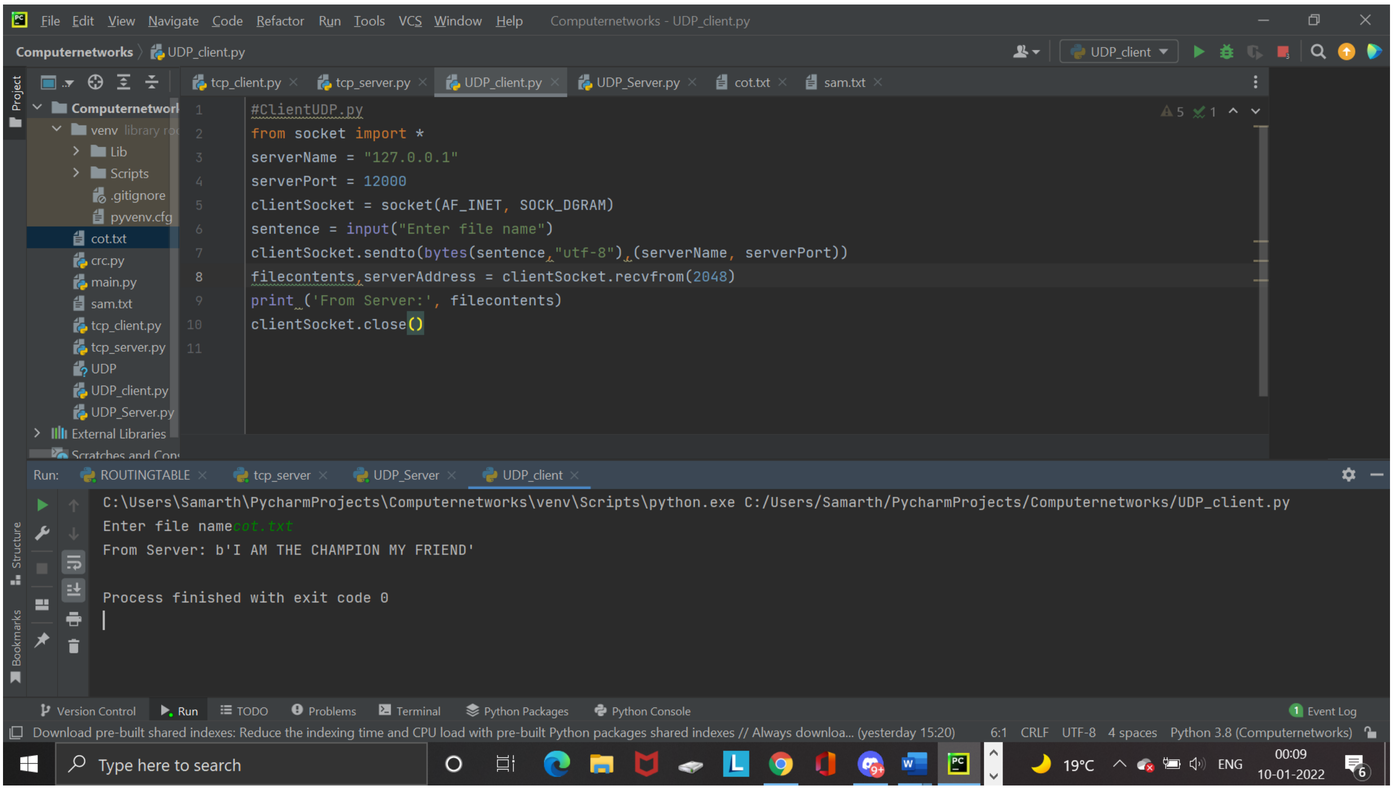
```
#ClientUDP.py
from socket import *
serverName="127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM) sentence =
input("Enter file name")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('From Server:', filecontents)
clientSocket.close()


#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)
        file=open(sentence,"r")
        l=file.read(2048)
        serverSocket.sendto(bytes(l,"utf-8"),clientAddress) print("sent
        back to client",l)
file.close()
```

OUTPUT:


Computernetworks - UDP_Server.py
Computernetworks
UDP_Server.py
tcp_client.py
tcp_server.py
UDP_client.py
UDP_Server.py
cot.txt
sam.txt
Run:
ROUTINGTABLE
tcp_server
UDP_Server
UDP_client
while 1
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always downloa... (yesterday 15:20)
4:1 CRLF UTF-8 4 spaces Python 3.8 (Computernetworks)
Type here to search
19°C ENG 00:09 10-01-2022

```
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()
```

```
C:\Users\Samarth\PycharmProjects\Computernetworks\venv\Scripts\python.exe C:/Users/Samarth/PycharmProjects/Computernetworks/UDP_Server.py
The server is ready to receive
cot.txt
```


Computernetworks - UDP_client.py
Computernetworks
UDP_client.py
tcp_client.py
tcp_server.py
UDP_client.py
UDP_Server.py
cot.txt
sam.txt
Run:
ROUTINGTABLE
tcp_server
UDP_Server
UDP_client
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always downloa... (yesterday 15:20)
6:1 CRLF UTF-8 4 spaces Python 3.8 (Computernetworks)
Type here to search
19°C ENG 00:09 10-01-2022

```
#ClientUDP.py
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()
```

```
C:\Users\Samarth\PycharmProjects\Computernetworks\venv\Scripts\python.exe C:/Users/Samarth/PycharmProjects/Computernetworks/UDP_client.py
Enter file namecot.txt
From Server: b'I AM THE CHAMPION MY FRIEND'

Process finished with exit code 0
```