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

```
import hashlib

def xor(a, b):

    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')

    return ''.join(result)


def mod2div(dividend, divisor):

    pick = len(divisor)
    tmp = dividend[0: pick]

    while pick < len(dividend):

        if tmp[0] == '1':
            tmp = xor(divisor, tmp) + dividend[pick]

        else:
            tmp = xor('0' * pick, tmp) + dividend[pick]
        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0' * pick, tmp)

    checkword = tmp
    return checkword


def encodeData(data, key):
    l_key = len(key)

    appended_data = data + '0' * (l_key - 1)
    remainder = mod2div(appended_data, key)

    codeword = data + remainder
    return codeword

def decodeData(code, key):
    remainder = mod2div(code, key)
    return remainder

data=input("Enter Data: ")
print("dataword:"+str(data))

key = "10001000000100001"
print("generating polynomial:"+key)
codeword = encodeData(data, key)
print("Checksum: ",codeword)
print("Transmitted Codeword:"+str(codeword))
```

```
code = input("enter transmitted codeword:")

recieved_data = int(decodeData(code, key))

if recieved_data == 0:
   print("NO ERROR")
else:
   print("ERROR")
   print(recieved_data)
```

2. Write a program for distance vector algorithm to find 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()
```

```
PS C:\Users\mvams\OneDrive\Desktop>  c:; cd 'c:\Users\mvams\OneDrive\Desktop'; & 'C:\Users\mvams\AppData\Local\P
ve\Desktop'; & 'C:\Users\mvams\AppData\Local\Programs\Python\Pythoe\extensions\ms-python.python-2021.12.1:\Users'
n310\python.exe' 'c:\Users\mvams\.vscode\extensions\ms-python.pyth:\Users\mvams\OneDrive\Desktop\2.py'
on-2021.12.1559732655\pythonFiles\lib\python\debugpy\launcher' '64
472' '--' 'c:\Users\mvams\OneDrive\Desktop\2.py'
Enter the no. of routers:
5
Enter the adjacency matrix : Enter 99 for infinity
0 1 5 99 99
5 3 0 4 99
99 99 4 0 2
99 9 99 2 0
0 1 5 99 99
Routing table for  0
Dest     Cost    Next Hop
0        0       0
1        1       1
2        1       1
3        1       1
4        1       1
Routing table for  1
Dest     Cost    Next Hop
0        0       2
1        0       1
2        0       2
3        0       2
4        0       2
Routing table for  2
Dest     Cost    Next Hop
0        0       3
1        1       3
2        0       2
3        0       3
4        0       3
Routing table for  3
Dest     Cost    Next Hop
0        0       4
1        1       4
2        1       4
3        0       3
4        0       4
Routing table for  4
Dest     Cost    Next Hop
0        0       0
1        1       1
2        1       1
3        1       1
4        0       4
PS C:\Users\mvams\OneDrive\Desktop>
```

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

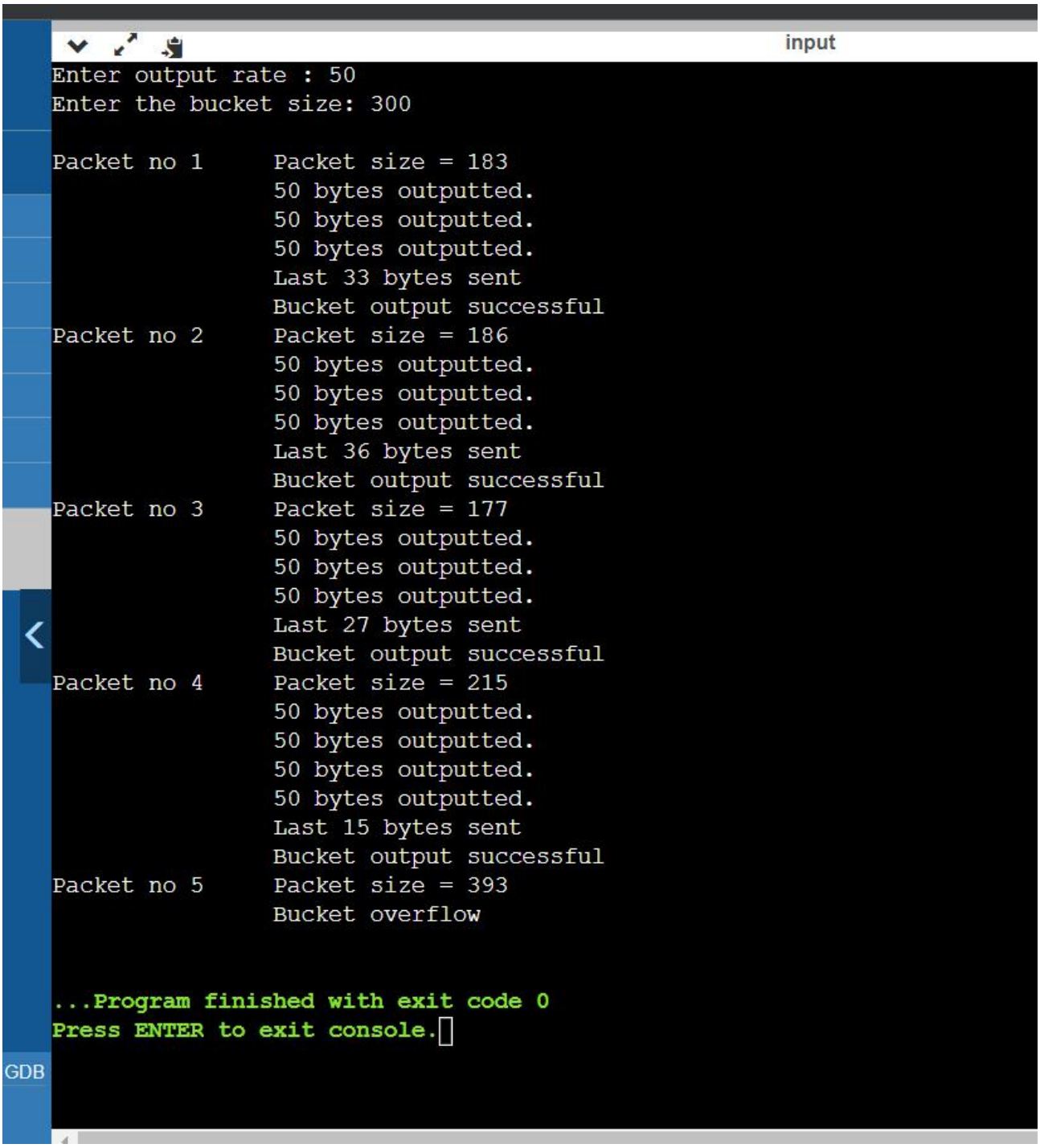
```
import sys

class Graph:
    def __init__(self,vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printSolution(self, dist):
        print("Vertex \tDistance from Source")
        for node in range(self.V):
```

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

    def minDistance(self, dist, sptSet):

        min = sys.maxsize

        for v in range(self.V):
            if dist[v] < min and sptSet[v] == False:
                min = dist[v]
                min_index = v

        return min_index

    def dijkstra(self, src):
        dist = [sys.maxsize] * self.V
        dist[src] = 0
        sptSet = [False] * self.V

        for cout in range(self.V):

            u = self.minDistance(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.printSolution(dist)

g = Graph(9)
g.graph = [ [0, 4, 0, 0, 0, 0, 0, 8, 0],
            [4, 0, 8, 0, 0, 0, 0, 11, 0],
            [0, 8, 0, 7, 0, 4, 0, 0, 2],
            [0, 0, 7, 0, 9, 14, 0, 0, 0],
            [0, 0, 0, 9, 0, 10, 0, 0, 0],
            [0, 0, 4, 14, 10, 0, 2, 0, 0],
            [0, 0, 0, 0, 0, 2, 0, 1, 6],
            [8, 11, 0, 0, 0, 0, 1, 0, 7],
            [0, 0, 2, 0, 0, 0, 6, 7, 0]
        ]

g.dijkstra(0)
```

```
PS C:\Users\mvams\OneDrive\Desktop>  c:; cd 'c:\Users\mvams\OneDrive\Desktop';
310\python.exe' 'c:\Users\mvams\.vscode\extensions\ms-python.python-2021.12.155
3' '--' 'c:\Users\mvams\OneDrive\Desktop\3.py'
Vertex  Distance from Source
0       0
1       4
2       12
3       19
4       21
5       11
6       9
7       8
8       14
PS C:\Users\mvams\OneDrive\Desktop>
```

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

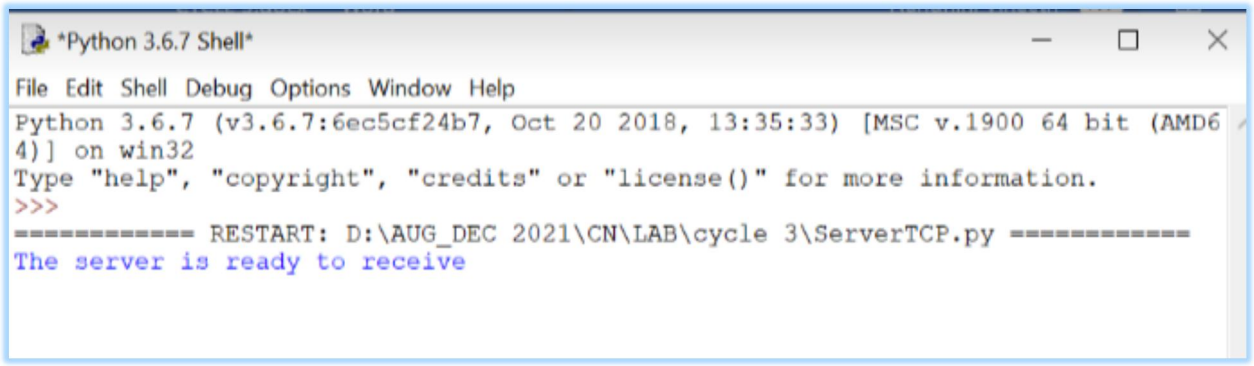
```
#include<bits/stdc++.h>
#include<unistd.h>
using namespace std;

int bucketSize;
void bucketInput(int a,int b)
{
	if(a > bucketSize)
		cout<<"\n\t\tBucket overflow";
	else{
		sleep(1);
		while(a > b){
			cout<<"\n\t\t"<<b<<" bytes outputted.";
			a-=b;
			sleep(1);
		}
		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;
   cout<<"Enter the bucket size: ";
   cin>>bucketSize;
	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;
}
```


input
Enter output rate : 50
Enter the bucket size: 300
Packet no 1 Packet size = 183
50 bytes outputted.
50 bytes outputted.
50 bytes outputted.
Last 33 bytes sent
Bucket output successful
Packet no 2 Packet size = 186
50 bytes outputted.
50 bytes outputted.
50 bytes outputted.
Last 36 bytes sent
Bucket output successful
Packet no 3 Packet size = 177
50 bytes outputted.
50 bytes outputted.
50 bytes outputted.
Last 27 bytes sent
Bucket output successful
Packet no 4 Packet size = 215
50 bytes outputted.
50 bytes outputted.
50 bytes outputted.
50 bytes outputted.
Last 15 bytes sent
Bucket output successful
Packet no 5 Packet size = 393
Bucket overflow
...Program finished with exit code 0
Press ENTER to exit console.
GDB

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.

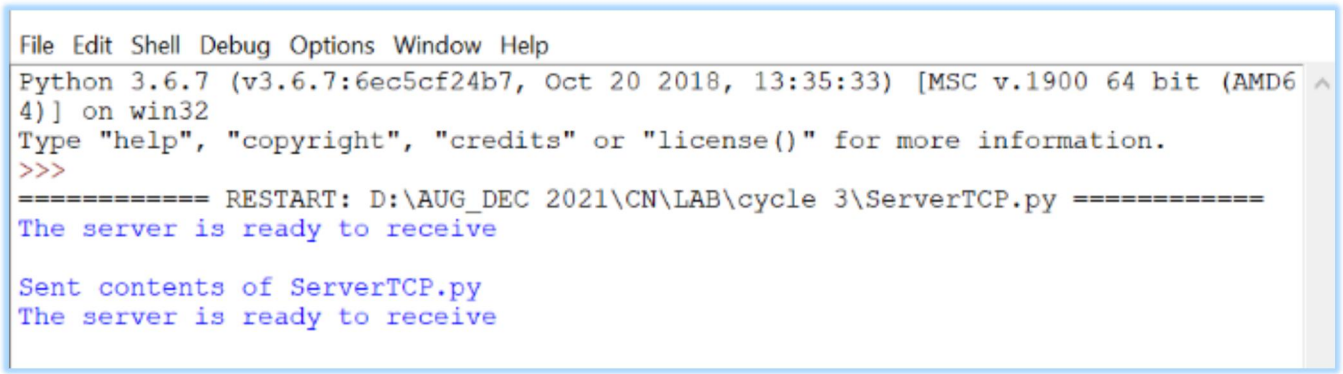
ClientTCP.py

```
from socket import *
serverName = &#39;127.0.0.1&#39;
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input(&quot;\nEnter file name: &quot;)
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print (&#39;\nFrom Server:\n&#39;)
print(filecontents)
clientSocket.close()
```



ServerTCP.py

```
from socket import *
serverName=&quot;127.0.0.1&quot;
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print (&quot;The server is ready to receive&quot;)
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file=open(sentence,&quot;r&quot;)
l=file.read(1024)
connectionSocket.send(l.encode())
print (&#39;\nSent contents of &#39; + sentence)
file.close()
connectionSocket.close()
```



10. 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.

ClientUDP.py

```
from socket import *
serverName = &amp;quot;127.0.0.1&amp;quot;
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input(&amp;quot;\nEnter file name: &amp;quot;)
clientSocket.sendto(bytes(sentence,&amp;quot;utf-8&amp;quot;),(serverName, serverPort))
filecontents,serverAddress = clientSocket.recvfrom(2048)
print (&amp;#39;\nReply from Server:\n&amp;#39;)
print (filecontents.decode(&amp;quot;utf-8&amp;quot;))
# for i in filecontents:
# print(str(i), end = &amp;#39;&amp;#39;)
clientSocket.close()
clientSocket.close()
```

ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind((&amp;quot;127.0.0.1&amp;quot;, serverPort))
print (&amp;quot;The server is ready to receive&amp;quot;)
while 1:
sentence, clientAddress = serverSocket.recvfrom(2048)
sentence = sentence.decode(&amp;quot;utf-8&amp;quot;)
file=open(sentence,&amp;quot;r&amp;quot;)
l=file.read(2048)
serverSocket.sendto(bytes(l,&amp;quot;utf-8&amp;quot;),clientAddress)
print (&amp;#39;\nSent contents of &amp;#39;, end = &amp;#39; &amp;#39;)
print (sentence)
# for i in sentence:
# print (str(i), end = &amp;#39;&amp;#39;)
file.close()
```