

CYCLE 2

Computer Networks Lab

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

Program :

```
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)

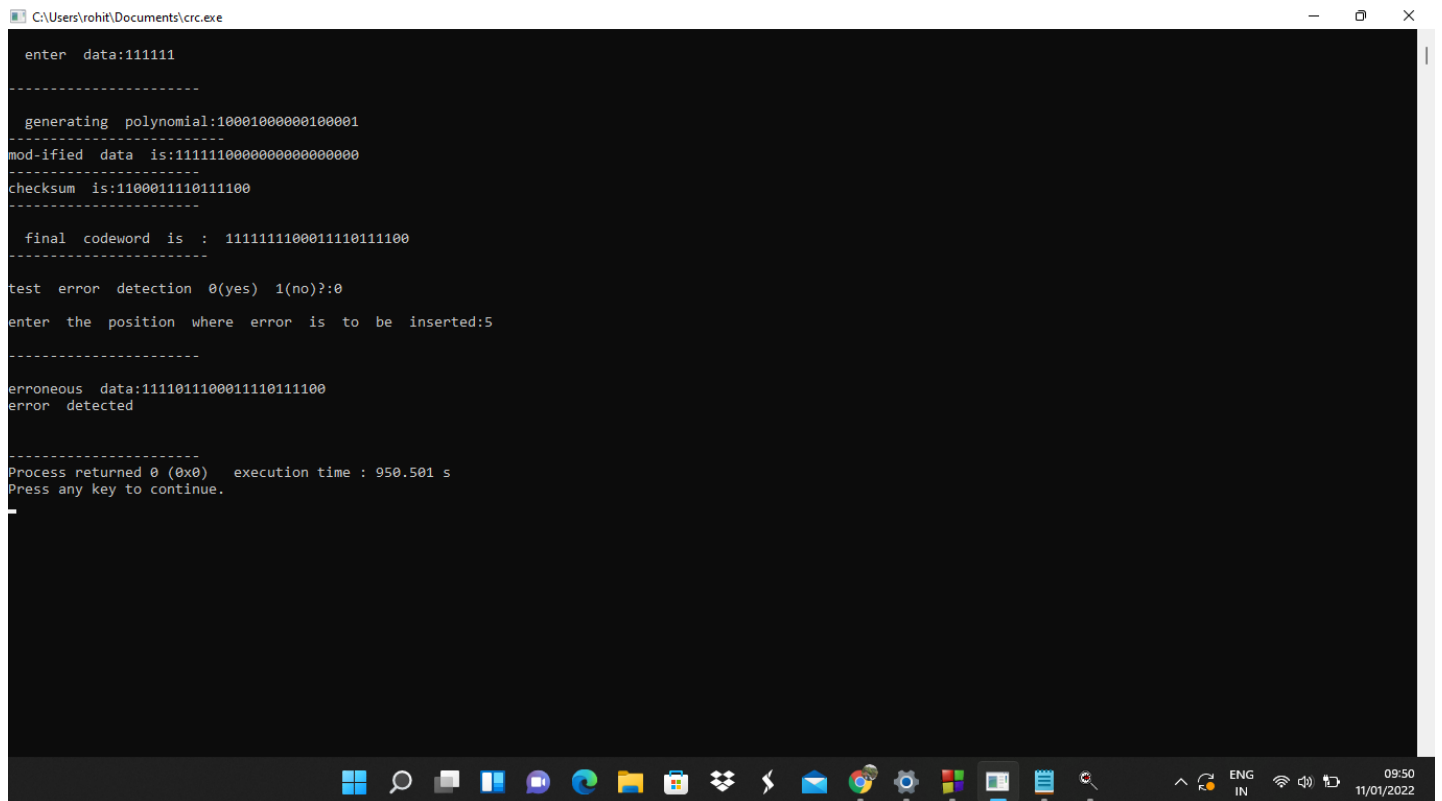
    # Checking the logic:

    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:



The screenshot shows a Windows command prompt window titled "C:\Users\rohit\Documents\crc.exe". The user enters "data:111111". The program then displays the following output:

```
-----
generating polynomial:10001000000100001
-----
mod-ified data is:1111110000000000000000
-----
checksum is:1100011110111100
-----
final codeword is : 111111100011110111100
-----
test error detection 0(yes) 1(no)?:0
enter the position where error is to be inserted:5
-----
erroneous data:1111011100011110111100
error detected
-----
Process returned 0 (0x0) execution time : 950.501 s
Press any key to continue.
```

The Windows taskbar at the bottom shows various application icons, including the Start button, Search, Task View, and several open applications like File Explorer, Microsoft Edge, and the Windows Store. The system tray on the right shows the date and time as 09:50 on 11/01/2022, along with icons for network, volume, and power.

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 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:

```
Enter the number of nodes : 5
```

```
Enter the cost matrix :
```

```
0 3 2 1 99
3 0 99 99 5
2 99 0 99 6
1 99 99 0 7
99 5 6 7 0
```

```
State value for router 1 is
```

```
node 1 via 1 Distance0
node 2 via 2 Distance3
node 3 via 3 Distance2
node 4 via 4 Distance1
node 5 via 2 Distance8
```

```
State value for router 2 is
```

```
node 1 via 1 Distance3
node 2 via 2 Distance0
node 3 via 1 Distance5
node 4 via 1 Distance4
node 5 via 5 Distance5
```

```
State value for router 3 is
```

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[i] = -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:

```

C:\Users\rohit\Documents\dijkstrasalgo.exe
Distance Matrix (5x5,):
0 3 1 99 99
3 0 7 5 1
1 7 0 2 99
99 5 2 0 7
99 1 99 7 0
Enter the source vertex: (0-4)
2
Vertex    Distance    Path
0 -> 1      4          0 0 1
0 -> 2      0          0
0 -> 3      2          0 3
0 -> 4      5          0 0 1 4
Process returned 0 (0x0)   execution time : 82.998 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:

ne_python_compiler



input

```
Enter Bucket Size: 512
Enter Output Rate: 100
Enter Input Packets: 3 33 133 1350
Packet No: 0 Packet Size: 3
    Last 3 bytes sent
    Bucket output successful

Packet No: 1 Packet Size: 33
    Last 33 bytes sent
    Bucket output successful

Packet No: 2 Packet Size: 133
    100 bytes sent
    Last 33 bytes sent
    Bucket output successful

Packet No: 3 Packet Size: 1350
    Bucket Overflow!!!

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

e_python_compiler



input

```
Enter Bucket Size: 8
Enter Output Rate: 2
Enter Input Packets: 8 2 5 6 3
Packet No: 0 Packet Size: 8
    2 bytes sent
    2 bytes sent
    2 bytes sent
    Last 2 bytes sent
    Bucket output successful

Packet No: 1 Packet Size: 2
    Last 2 bytes sent
    Bucket output successful

Packet No: 2 Packet Size: 5
    2 bytes sent
    2 bytes sent
    Last 1 bytes sent
    Bucket output successful

Packet No: 3 Packet Size: 6
    2 bytes sent
    2 bytes sent
    Last 2 bytes sent
    Bucket output successful

Packet No: 4 Packet Size: 3
    2 bytes sent
    Last 1 bytes sent
    Bucket output successful

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

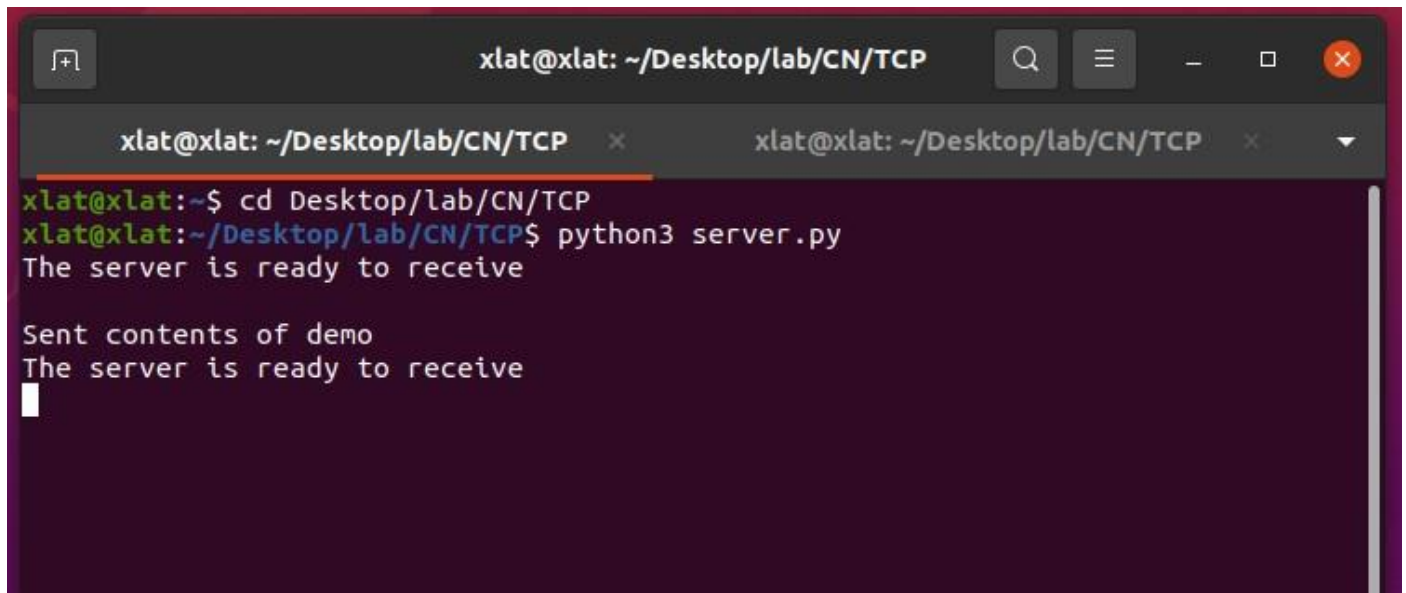
5 Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

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()
```

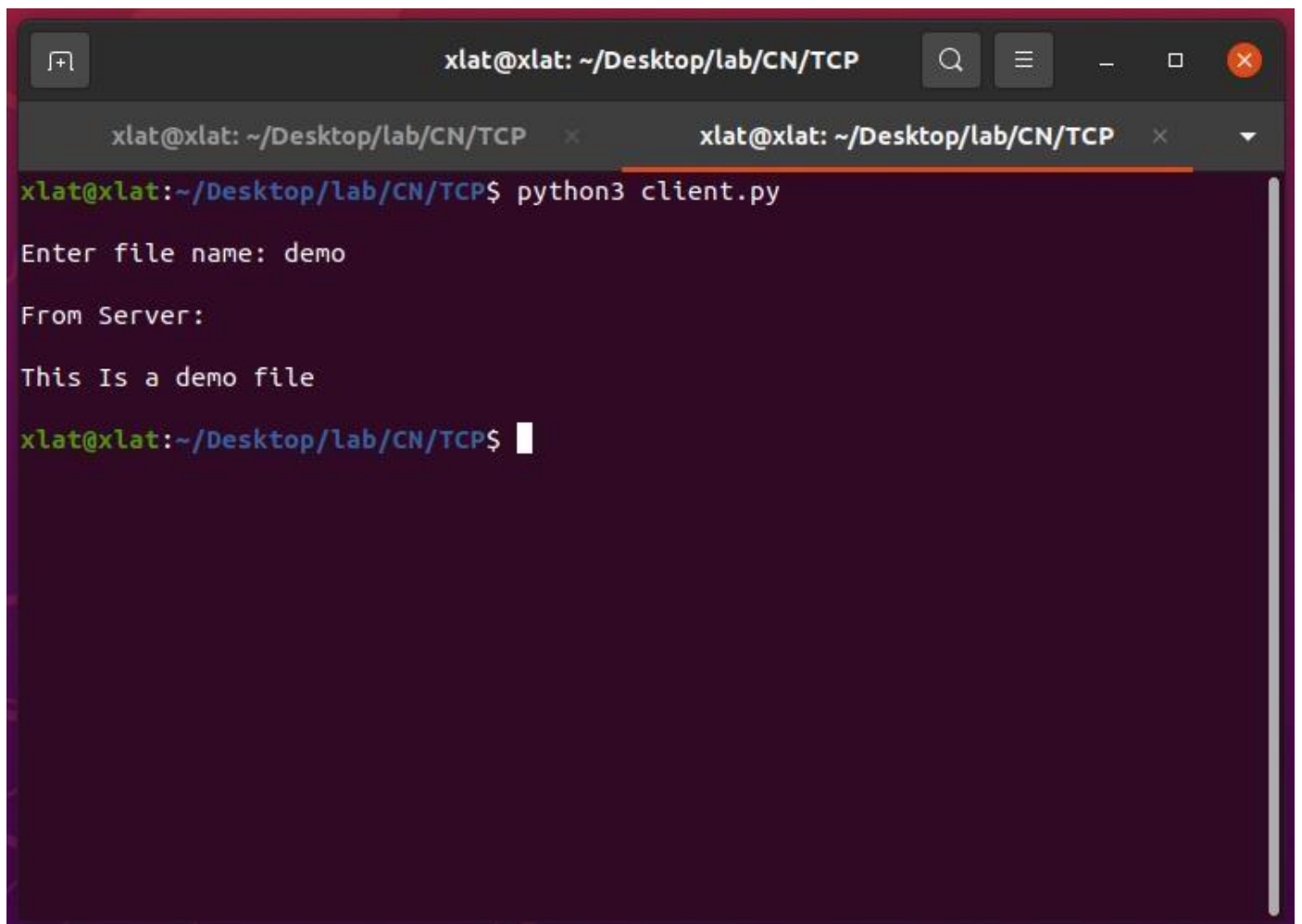
Output:



```
xlat@xlat: ~/Desktop/lab/CN/TCP
xlat@xlat:~/Desktop/lab/CN/TCP$ cd Desktop/lab/CN/TCP
xlat@xlat:~/Desktop/lab/CN/TCP$ python3 server.py
The server is ready to receive

Sent contents of demo
The server is ready to receive
█
```

A terminal window titled 'xlat@xlat: ~/Desktop/lab/CN/TCP' with two tabs. The first tab is active and shows the execution of 'python3 server.py'. The output indicates the server is ready to receive, then receives 'Sent contents of demo', and then returns to 'The server is ready to receive' with a cursor on the next line.



```
xlat@xlat: ~/Desktop/lab/CN/TCP
xlat@xlat:~/Desktop/lab/CN/TCP$ python3 client.py
Enter file name: demo
From Server:
This Is a demo file
xlat@xlat:~/Desktop/lab/CN/TCP$ █
```

A terminal window titled 'xlat@xlat: ~/Desktop/lab/CN/TCP' with two tabs. The second tab is active and shows the execution of 'python3 client.py'. The user enters 'demo' as the file name. The output shows 'From Server:' followed by 'This Is a demo file'. The prompt returns to 'xlat@xlat:~/Desktop/lab/CN/TCP\$' with a cursor.

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:

```
C:\CN-LAB\Scripts\python.exe C:/Users/Dell/PycharmProjects/CN-LAB/main2.py
The server is ready to receive
SENT BACK TO CLIENT test.html is thr okji
|
```

The server.py is executed first to set up server..and file name is passed

```
main2 x  main x
C:\CN-LAB\Scripts\python.exe C:/Users/Dell/PycharmProjects/CN-LAB/main.py
Enter file name test.html
From Server: <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>hello ji</title>
</head>
<body>

</body>
</html>

Process finished with exit code 0
|
```