CN Lab-Cycle 2

Shashank Sharma
1BM19CS147

1 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)

    # 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("Enterthedatapolynomial:\n")
data = polytobin(string)
print(key, data)
encode(data, key)
```

Output:

```
Enter the key polynomial:
x16+x12+x4+1
[16, 12, 4, 0]
10001000000010001
Enter the data polynomial:
x15+x12+x11+x8+x7+x4+x3+1
[15, 12, 11, 8, 7, 4, 3, 0]
1001100110011001
10001000000010001 1001100110011001
remaindar=00001001000010010
code=10011001100110010000100100001001 0
Remaindar we get when we do not have error=00000000000000000
Remaindar we get when we have error=00110011001100000

Process finished with exit code 0
```

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()
```

Output:

```
Enter the no. of routers:
3
Enter the adjacency matrix : Enter 99 for infinity
0 1 999
1 0 1
999 1 0
Routing table for  0
Dest     Cost    Next Hop
0    0   0
1    1   1
2    2   1
Routing table for  1
Dest     Cost    Next Hop
0    1   0
1    0   1
2    1   2
Routing table for  2
Dest     Cost    Next Hop
0    2   1
1    1   1
2    0   2

Process finished with exit code 0
```

3 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:

```
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.

```
#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:

```
Enter output rate : 108

Packet no 1     Packet size = 267
                108 bytes outputted.
                108 bytes outputted.
                Last 51 bytes sent
                Bucket output successful
Packet no 2     Packet size = 600
                Bucket overflow
Packet no 3     Packet size = 324
                108 bytes outputted.
                108 bytes outputted.
                Last 108 bytes sent
                Bucket output successful
Packet no 4     Packet size = 658
                Bucket overflow
Packet no 5     Packet size = 664
                Bucket overflow

Process returned 0 (0x0)   execution time : 58.068 s
```

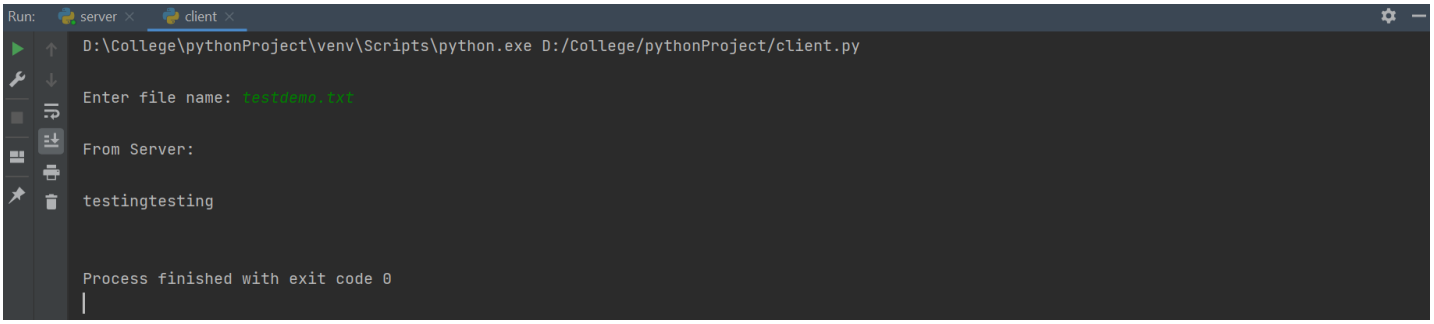
5 Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

```
#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:


Run:
server
client
D:\College\pythonProject\venv\Scripts\python.exe D:\College\pythonProject\server.py
The server is ready to receive
Sent contents of testdemo.txt
The server is ready to receive

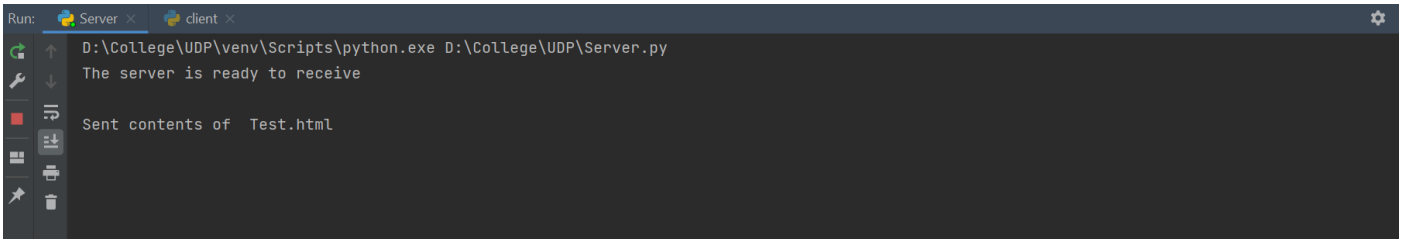

Run:
server
client
D:\College\pythonProject\venv\Scripts\python.exe D:/College/pythonProject/client.py
Enter file name: testdemo.txt
From Server:
testingtesting
Process finished with exit code 0

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.

```
 #ClientUDP.py
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)

sentence = input("\nEnter file name:  ")

clientSocket.sendto(bytes(sentence,"utf-8"),(serverName, serverPort))

filecontents,serverAddress = clientSocket.recvfrom(2048)
print ('\nReply from Server:\n')
print (filecontents.decode("utf-8"))
# for i in filecontents:
   # print(str(i), end = '')
clientSocket.close()
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)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    l=file.read(2048)

    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)

    print ('\nSent contents of ', end = ' ')
    print (sentence)
   #  for i in sentence:
      #  print (str(i), end = '')
    file.close()
```

Output:



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