

Name: Naveena KN

USN: 1BM20CS411

CN LAB CYCLE 2 PROGRAMS

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

```
import java.util.*;

public class CRC{

public static int n;

public static void main(String[] args){

    Scanner in=new Scanner(System.in);

    CRC ob=new CRC();

    String code, copy, rec,zero="0000000000000000";

    System.out.print("Enter poly: ");

    code=in.nextLine();

    System.out.println("Generating polynomial: 10001000000100001");

    n=code.length()

    copy=code;

    code+=zero;

    System.out.println("Modified poly: "+code);

    code=ob.divide(code);

    System.out.println("Checksum: "+code.substring(n));

    copy=copy.substring(0,n)+code.substring(n);

    System.out.println("Final Codeword: "+copy);

    // System.out.print("\nEnter recieved data: ");

    // rec=in.nextLine();

    // if(zero.equals(ob.divide(rec).substring(n)))

    //     System.out.println("Correct bits recieved");

    // else

    //     System.out.println("Recieved frame contains one or more errors");

    System.out.print("Test Error detection 0(yes) 1(no)? : ");

    int choice = in.nextInt();
```

```

if(choice == 0){
    System.out.print("Enter position on error: ");
    int errorPos = in.nextInt();
    if(copy.charAt(errorPos) == '1')
        copy = copy.substring(0,errorPos) + "0" + copy.substring(errorPos+1);
    else
        copy = copy.substring(0,errorPos) + "1" + copy.substring(errorPos+1);
    System.out.println("Errorneous data: "+copy);
    System.out.println("Error detected");
    }
    else
        System.out.println("No Error detection");
    }
    public String divide(String s){
        int i,j;
        char x;
        String div="100010000000100001";
        for(i=0;i<n;i++){
            x=s.charAt(i);
            for(j=0;j<17;j++){
                if(x=='1'){
                    if(s.charAt(i+j)!=div.charAt(j))
                        s=s.substring(0,i+j)+"1"+s.substring(i+j+1);
                    else
                        s=s.substring(0,i+j)+"0"+s.substring(i+j+1);
                }
            }
        }
        return s;
    }
}

```

OUTPUT:

```
C:\Users\NAVEENA\Desktop>java CRC
Enter poly: 1011101
Generating polynomial: 10001000000100001
Modified poly: 10111010000000000000000
Checksum: 1000101101011000
Final Codeword: 10111011000101101011000
Test Error detection 0(yes) 1(no)? : 0
Enter position on error: 2
Errorneous data: 10011011000101101011000
Error detected

C:\Users\NAVEENA\Desktop>java CRC
Enter poly: 1011101
Generating polynomial: 10001000000100001
Modified poly: 10111010000000000000000
Checksum: 1000101101011000
Final Codeword: 10111011000101101011000
Test Error detection 0(yes) 1(no)? : 1
No Error detection
```



```
graph[i][j] = 9999;
}

for(int i = 0; i < e; i++)
{
    System.out.println("Please enter data for Edge " + (i + 1) + ":");
    System.out.print("Source: ");
    int s = Integer.parseInt(br.readLine());
    s--;
    System.out.print("Destination: ");
    int d = Integer.parseInt(br.readLine());
    d--;
    System.out.print("Cost: ");
    int c = Integer.parseInt(br.readLine());
    graph[s][d] = c;
    graph[d][s] = c;
}

dvr_calc_disp("The initial Routing Tables are: ");

System.out.print("Please enter the Source Node for the edge whose cost has changed: ");
int s = Integer.parseInt(br.readLine());
s--;
System.out.print("Please enter the Destination Node for the edge whose cost has changed: ");
int d = Integer.parseInt(br.readLine());
d--;
System.out.print("Please enter the new cost: ");
int c = Integer.parseInt(br.readLine());
graph[s][d] = c;
graph[d][s] = c;
```

```
dvr_calc_disp("The new Routing Tables are: ");  
}
```

```
static void dvr_calc_disp(String message)  
{  
    System.out.println();  
    init_tables();  
    update_tables();  
    System.out.println(message);  
    print_tables();  
    System.out.println();  
}
```

```
static void update_table(int source)  
{  
    for(int i = 0; i < v; i++)  
    {  
        if(graph[source][i] != 9999)  
        {  
            int dist = graph[source][i];  
            for(int j = 0; j < v; j++)  
            {  
                int inter_dist = rt[i][j];  
                if(via[i][j] == source)  
                    inter_dist = 9999;  
                if(dist + inter_dist < rt[source][j])  
                {  
                    rt[source][j] = dist + inter_dist;  
                    via[source][j] = i;  
                }  
            }  
        }  
    }  
}
```

```
}
```

```
static void update_tables()
```

```
{
```

```
    int k = 0;
```

```
    for(int i = 0; i < 4*v; i++)
```

```
    {
```

```
        update_table(k);
```

```
        k++;
```

```
        if(k == v)
```

```
            k = 0;
```

```
    }
```

```
}
```

```
static void init_tables()
```

```
{
```

```
    for(int i = 0; i < v; i++)
```

```
    {
```

```
        for(int j = 0; j < v; j++)
```

```
        {
```

```
            if(i == j)
```

```
            {
```

```
                rt[i][j] = 0;
```

```
                via[i][j] = i;
```

```
            }
```

```
        else
```

```
        {
```

```
            rt[i][j] = 9999;
```

```
            via[i][j] = 100;
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```

static void print_tables()
{
    for(int i = 0; i < v; i++)
    {
        for(int j = 0; j < v; j++)
        {
            System.out.print("Dist: " + rt[i][j] + "  ");
        }
        System.out.println();
    }
}

```

OUTPUT:

```

C:\Users\NAVEENA\Desktop>java Main
Please enter the number of Vertices:
5
Please enter the number of Edges:
0

The initial Routing Tables are:
Dist: 0    Dist: 9999    Dist: 9999    Dist: 9999    Dist: 9999
Dist: 9999    Dist: 0    Dist: 9999    Dist: 9999    Dist: 9999
Dist: 9999    Dist: 9999    Dist: 0    Dist: 9999    Dist: 9999
Dist: 9999    Dist: 9999    Dist: 9999    Dist: 0    Dist: 9999
Dist: 9999    Dist: 9999    Dist: 9999    Dist: 9999    Dist: 0

Please enter the Source Node for the edge whose cost has changed: 1
Please enter the Destination Node for the edge whose cost has changed: 2
Please enter the new cost: 3

The new Routing Tables are:
Dist: 0    Dist: 3    Dist: 9999    Dist: 9999    Dist: 9999
Dist: 3    Dist: 0    Dist: 9999    Dist: 9999    Dist: 9999
Dist: 9999    Dist: 9999    Dist: 0    Dist: 9999    Dist: 9999
Dist: 9999    Dist: 9999    Dist: 9999    Dist: 0    Dist: 9999
Dist: 9999    Dist: 9999    Dist: 9999    Dist: 9999    Dist: 0

```


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

```
import java.util.*;
```

```
class Edge{  
    int src, dest, w;  
    public Edge(int src, int dest, int w){  
        this.src = src;  
        this.dest = dest;  
        this.w = w;  
    }  
}
```

```
class Node {  
    int vertex, w;  
    public Node(int vertex, int w) {  
        this.vertex = vertex;  
        this.w = w;  
    }  
}
```

```
class Graph{  
    List<List<Edge>> edgeList = null;  
    Graph(List<Edge> edges, int N){  
        edgeList = new ArrayList<>();  
        for (int i = 0; i < N; i++) {  
            edgeList.add(new ArrayList<>());  
        }  
        for (Edge edge: edges){  
            edgeList.get(edge.src).add(edge);  
        }  
    }  
}
```

```

    }
}
}

```

```

class Dijkstra{
    private static void getPath(int[] prev, int i, List<Integer> route){
        if (i >= 0){
            getPath(prev, prev[i], route);
            route.add(i);
        }
    }

    public static void getShortestPath(Graph graph, int src, int N){
        PriorityQueue<Node> minHeap;
        minHeap = new PriorityQueue<>(Comparator.comparingInt(node -> node.w));
        minHeap.add(new Node(src, 0));
        List<Integer> dist = new ArrayList<>(Collections.nCopies(N, Integer.MAX_VALUE));
        dist.set(src, 0);
        boolean[] done = new boolean[N];
        done[src] = true;
        int[] prev = new int[N];
        prev[src] = -1;
        List<Integer> route = new ArrayList<>();
        while (!minHeap.isEmpty()){
            Node node = minHeap.poll();
            int u = node.vertex;
            for (Edge edge: graph.edgeList.get(u)){
                int v = edge.dest;
                int w = edge.w;
                if (!done[v] && (dist.get(u) + w) < dist.get(v)){
                    dist.set(v, dist.get(u) + w);
                    prev[v] = u;
                }
            }
        }
    }
}

```

```

        minHeap.add(new Node(v, dist.get(v)));
    }
}
done[u] = true;
}

for(int i = 1; i < N; ++i){
    if (i != src && dist.get(i) != Integer.MAX_VALUE) {
        getPath(prev, i, route);
        System.out.printf("Route is %d ==> %d and min cost = %d and path is %s\n",
            src, i, dist.get(i), route);
        route.clear();
    }
}
}

public static void main(String[] args){
    Scanner s = new Scanner(System.in);
    List<Edge> edges = new ArrayList<>();
    System.out.println("Enter number of vertices");
    int n = s.nextInt();
    System.out.println("Enter the adjacency weighted matrix");
    int[][] mat = new int[n][n];
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            mat[i][j] = s.nextInt();
        }
    }

    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            if(i == j) continue;

```

```

        if(mat[i][j] != -1){
            edges.add(new Edge(i, j, mat[i][j]));
        }
    }
}

Graph graph = new Graph(edges, n);
int src = 0;
getShortestPath(graph, src, n);
s.close();
}
}

```

OUTPUT:

```

C:\Users\NAVEENA\Desktop>java Dijkstra
Enter number of vertices
5
Enter the adjacency weighted matrix
-1 10 -1 -1 3
-1 -1 2 -1 4
-1 -1 -1 9 -1
-1 -1 7 -1 -1
-1 1 8 2 -1
Route is 0 => 1 and min cost = 4 and path is [0, 4, 1]
Route is 0 => 2 and min cost = 6 and path is [0, 4, 1, 2]
Route is 0 => 3 and min cost = 5 and path is [0, 4, 3]
Route is 0 => 4 and min cost = 3 and path is [0, 4]

```

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

```
#include<stdio.h>
```

```
    #include<stdlib.h>
```

```
    #include<unistd.h>
```

```
    #define NOF_PACKETS 5
```

```
    /*
```

```
    int rand (int a)
```

```
    {
```

```
    int rn = (random() % 10) % a;
```

```
    return rn == 0 ? 1 : rn;
```

```
    }
```

```
    */
```

```
    /*
```

```
    #include <stdlib.h>;
```

```
    long int random(void);
```

```
3
```

The random() function uses a nonlinear additive feedback random number

generator employing a default ta-

ble of size 31 long integers to return successive pseudo-random numbers in the

range from 0 to RAND_MAX.

The period of this random number generator is very large, approximately $16 * ((2^{31}) - 1)$.

```
((2^31) - 1).
```

```
    */
```

```
    int main()
```

```
    {
```

```
    int packet_sz[NOF_PACKETS], i, clk, b_size, o_rate, p_sz_rm=0, p_sz, p_time, op;
```

```
    for(i = 0; i<NOF_PACKETS; ++i)
```

```
    packet_sz[i] = random() % 100;
```

```
    for(i = 0; i<NOF_PACKETS; ++i)
```

```
    printf("\npacket[%d]:%d bytes\t", i, packet_sz[i]);
```

```
    printf("\nEnter the Output rate:");
```

```

scanf("%d", &o_rate);

printf("Enter the Bucket Size:");

scanf("%d", &b_size);

for(i = 0; i<NOF_PACKETS; ++i)
{
    if( (packet_sz[i] + p_sz_rm) > b_size)
        if(packet_sz[i] > b_size)/*compare the packet siz with bucket size*/
            printf("\n\nIncoming packet size (%dbytes) is Greater than bucket capacity
(%dbytes)-PACKET REJECTED", packet_sz[i], b_size);
        else
            printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");
        else
        {
            p_sz_rm += packet_sz[i];
            printf("\n\nIncoming Packet size: %d", packet_sz[i]);
            printf("\nBytes remaining to Transmit: %d", p_sz_rm);
            //p_time = random() * 10;
            //printf("&quot;\nTime left for transmission: %d units&quot;;, p_time);
            //for(clk = 10; clk &lt;= p_time; clk += 10)
            while(p_sz_rm>0)
            {
                sleep(1);
                if(p_sz_rm)
                {
                    if(p_sz_rm <= o_rate)/*packet size remaining comparing with output rate*/
                        op = p_sz_rm, p_sz_rm = 0;
                    else
                        op = o_rate, p_sz_rm -= o_rate;
                    printf("\nPacket of size %d Transmitted", op);
                    printf("----Bytes Remaining to Transmit: %d", p_sz_rm);
                }
            }
            else

```

```

{
printf("\nNo packets to transmit!!");
}
}
}
}}

```

OUTPUT:

```

packet[0]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:35
Enter the Bucket Size:85

Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 48
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 13
Packet of size 13 Transmitted----Bytes Remaining to Transmit: 0

Incoming packet size (86bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED

Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 42
Packet of size 35 Transmitted----Bytes Remaining to Transmit: 7
Packet of size 7 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 15

```

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:

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name: ")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
6
print ('\nFrom Server:\n')
print(filecontents)
clientSocket.close()
```

Server:

```
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print ("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()

    file=open(sentence,"r")
    l=file.read(1024)
```



```
connectionSocket.send(l.encode())  
print ("\nSent contents of ' + sentence)  
file.close()  
connectionSocket.close()
```

OUTPUT:

Enter file name: ServerTCP.py

From Server:

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

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.

Client:

```
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()
```

Server:

```
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 = ' ')
```

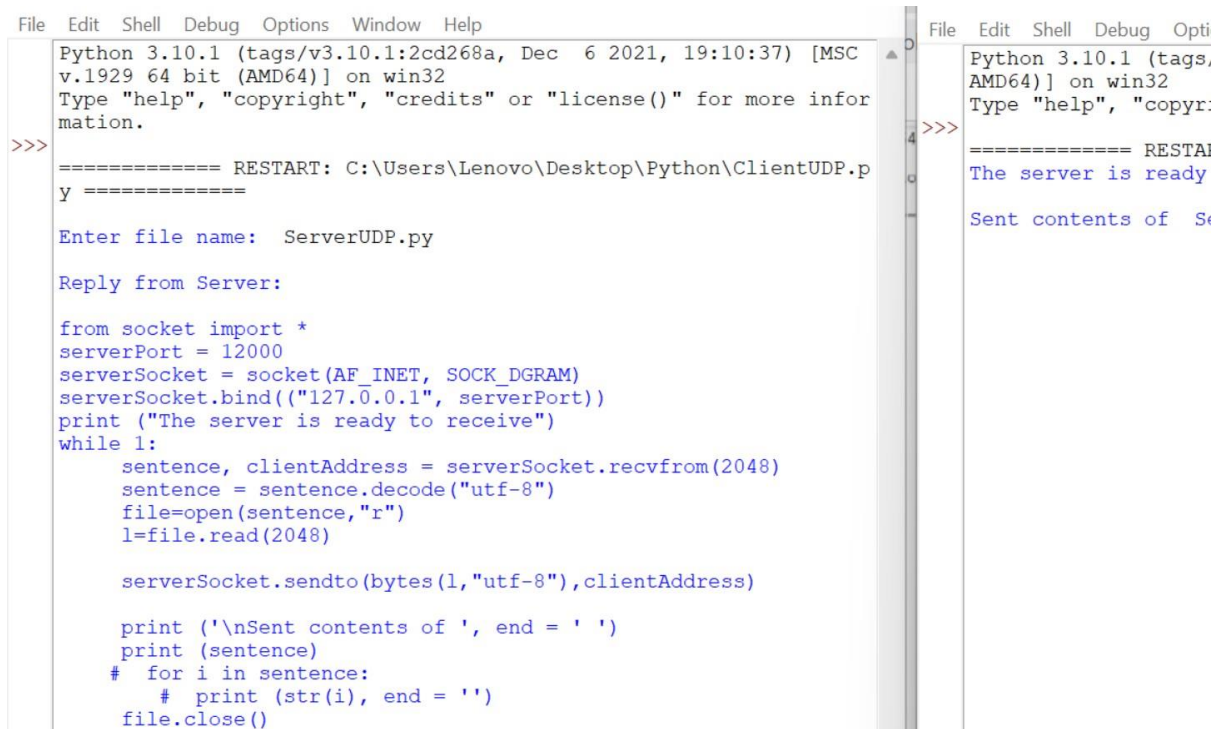
```
print (sentence)

# for i in sentence:

# print (str(i), end = '')

file.close()
```

OUTPUT:



The image shows two side-by-side Python IDE windows. The left window displays the source code for a UDP server, and the right window shows the output of the program.

```
File Edit Shell Debug Options Window Help
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC
v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
mation.
>>>
===== RESTART: C:\Users\Lenovo\Desktop\Python\ClientUDP.p
y =====
Enter file name: ServerUDP.py
Reply from Server:
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()
```

```
File Edit Shell Debug Opti
Python 3.10.1 (tags,
AMD64)] on win32
Type "help", "copyr:
>>>
===== RESTAI
The server is ready
Sent contents of S:
```