CN LAB RECORD - CYCLE 2

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

```
#include <iostream>
#include <string.h>
using namespace std;
int crc(char *ip, char *op, char *poly, int mode){
  strcpy(op, ip);
  if(mode){
    for(int i = 1; i < strlen(poly); i++)
      strcat(op, "0");
    }
  for(int i = 0; i < strlen(ip); i++){
    if(op[i] == '1'){
      for(int j = 0; j < strlen(poly); j++){
        if (op[i + j] == poly[j])
          op[i + j] = '0';
        else
          op[i + j] = '1';
      }
    }
  }
  for (int i = 0; i < strlen(op); i++)
    if (op[i] == '1')
      return 0;
    return 1;
  }

int main(){
  char ip[50], op[50], recv[50];
  char poly[] = "10001000000100001";
  cout << "Enter the input message in binary"<< endl;
  cin >> ip;
  crc(ip, op, poly, 1);
  cout << "The transmitted message is: " << ip << op + strlen(ip) << endl;
  cout << "Enter the recevied message in binary" << endl;
  cin >> recv;
  if(crc(recv, op, poly, 0))
    cout << "No error in data" << endl;
  else
    cout << "Error in data transmission has occurred" << endl;
```

```
  return 0;
}
```

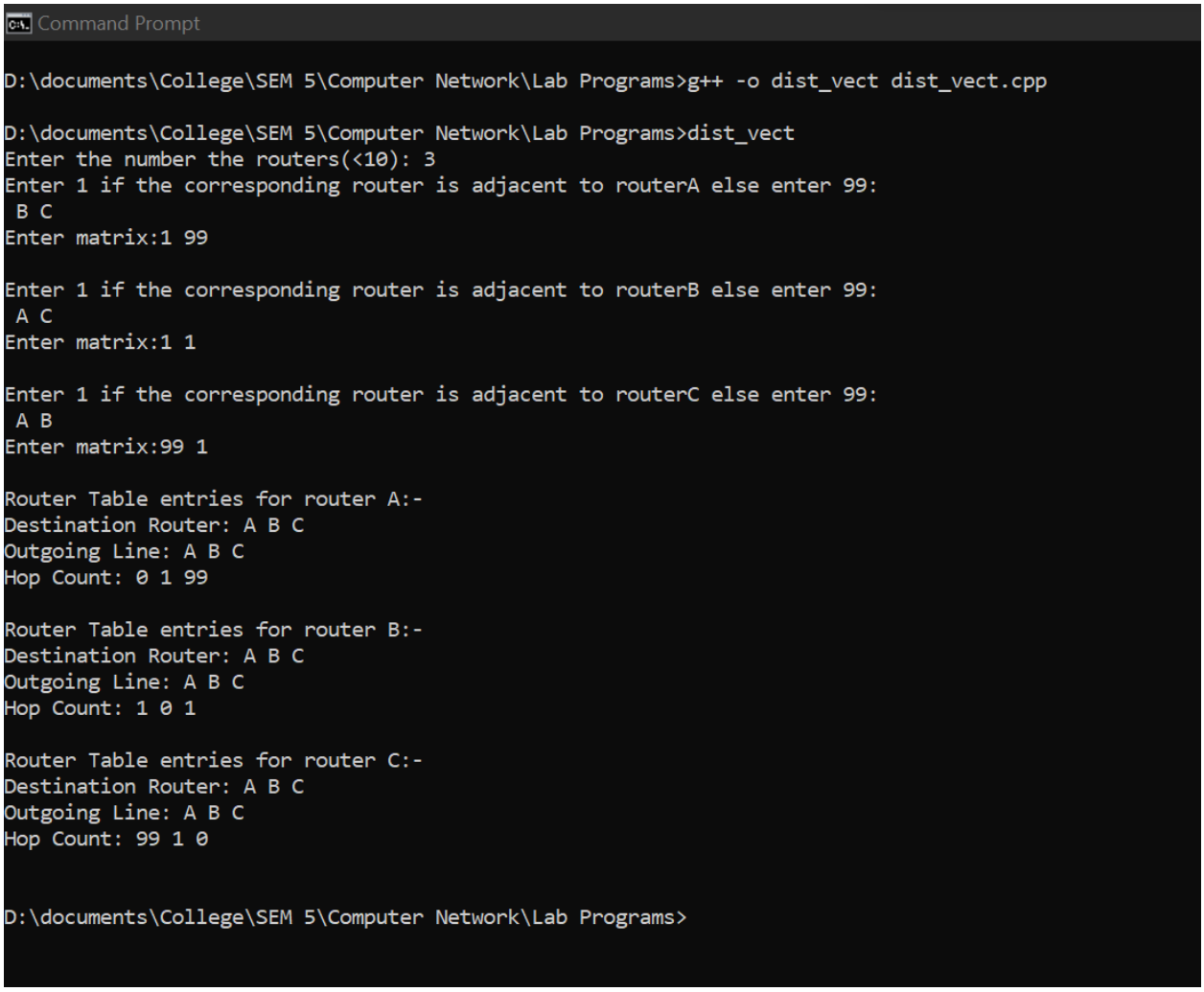
```
D:\documents\College\SEM 5\Computer Network\Lab Programs>crc
Enter the input message in binary
11111
The transmitted message is: 111111110001111011110
Enter the recevied message in binary
11111
No error in data

D:\documents\College\SEM 5\Computer Network\Lab Programs>crc
Enter the input message in binary
10110
The transmitted message is: 101100111001011110111
Enter the recevied message in binary
10111
Error in data transmission has occurred

D:\documents\College\SEM 5\Computer Network\Lab Programs>
```

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

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 10
int n;
class router{
  char adj_new[MAX], adj_old[MAX];
  int table_new[MAX], table_old[MAX];
  public:
    router( ){
      for(int i=0;i<MAX;i++)
        table_old[i]=table_new[i]=99;
      }
  void copy( ){
    for(int i=0;i<n;i++) {
      adj_old[i] =adj_new[i];
      table_old[i]=table_new[i];
    }
  }
  int equal( ){
    for(int i=0;i<n;i++)
    if(table_old[i]!=table_new[i]||adj_new[i]!=adj_old[i])
      return 0;
```

```
    return 1;
  }
  void input(int j){
    cout<<"Enter 1 if the corresponding router is adjacent to router"<<(char)('A'+j)<<" else enter
99: "<<endl<<" ";
    for(int i=0;i<n;i++)
      if(i!=j)
        cout<<(char)('A'+i)<<" ";
      cout<<"\nEnter matrix:";
      for(int i=0;i<n;i++){
        if(i==j)
          table_new[i]=0;
        else
          cin>>table_new[i];
        adj_new[i]= (char)('A'+i);
      }
    cout<<endl;
  }

void display(){
  cout<<"\nDestination Router: ";
  for(int i=0;i<n;i++)
    cout<<(char)('A'+i)<<" ";
  cout<<"\nOutgoing Line: ";
  for(int i=0;i<n;i++)
    cout<<adj_new[i]<<" ";
  cout<<"\nHop Count: ";
  for(int i=0;i<n;i++)
    cout<<table_new[i]<<" ";
  }

void build(int j){
  for(int i=0;i<n;i++)
    for(int k=0;(i!=j)&&(k<n);k++)
      if(table_old[i]!=99)
        if((table_new[i]+table_new[k])<table_new[k]) {
          table_new[k]=table_new[i]+table_new[k];
          adj_new[k]=(char)('A'+i);
        }
      }
    }
    r[MAX];

void build_table(){
```

```
  int i=0, j=0;
  while(i!=n){
    for(i=j;i<n;i++){
      r[i].copy();
      r[i].build(i);
    }
    for(i=0;i<n;i++)
      if(!r[i].equal()){
        j=i;
        break;
      }
    }
  }

int main(){
  cout<<"Enter the number the routers(<"<<MAX<<"): "; cin>>n;
  for(int i=0;i<n;i++) r[i].input(i);
    build_table();
    for(int i=0;i<n;i++) {
      cout<<"Router Table entries for router "<<(char)('A'+i)<<":-";
      r[i].display();
      cout<<endl<<endl;
    }
}
```



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

```
#include<bits/stdc++.h>
using namespace std;
#define V 3

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 \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<<"Please Enter The Graph (!!! Use 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 vertex: "<<endl;
  int src;
  cin>>src;
  dijkstra(graph, src);
```

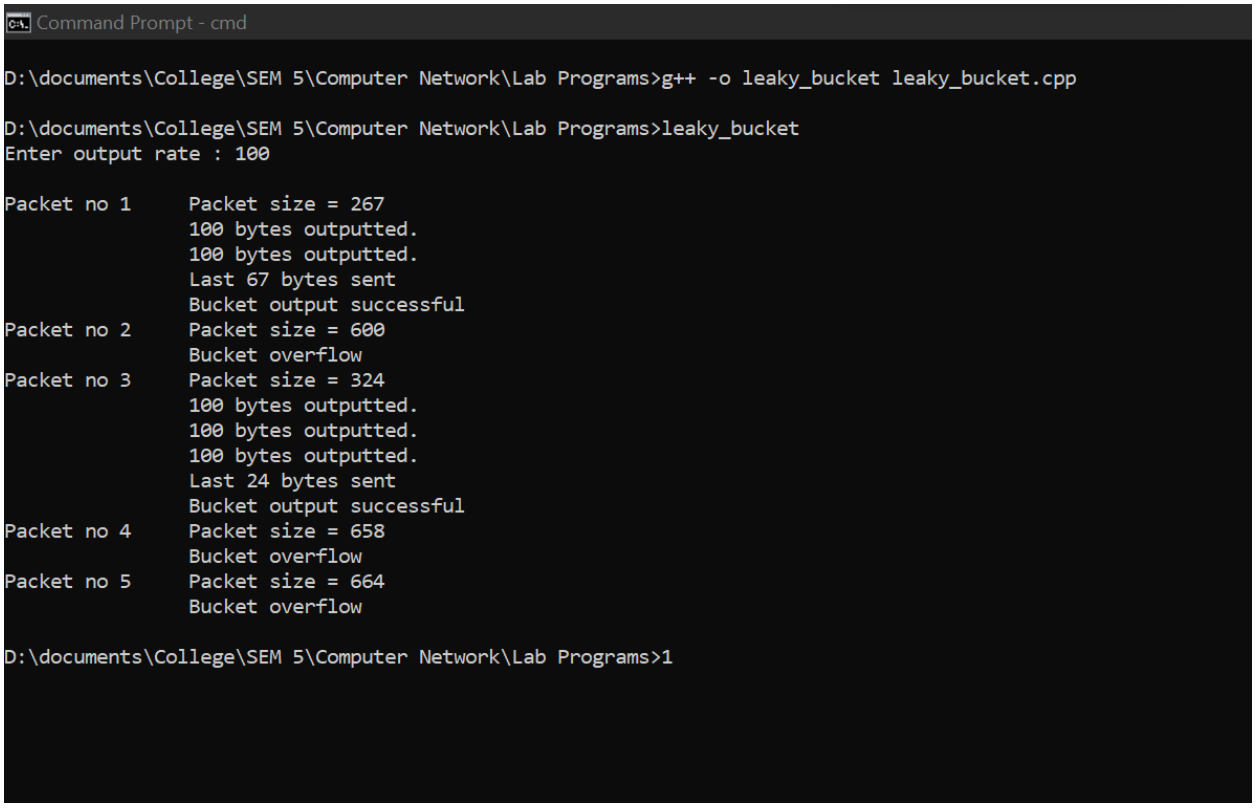
```
  cout<<endl;
  return 0;
}
```

```
D:\documents\College\SEM 5\Computer Network\Lab Programs>g++ -o Dijkstra's_algo Dijkstra's_algo.cpp

D:\documents\College\SEM 5\Computer Network\Lab Programs>Dijkstra's_algo
Please Enter The Graph (!!! Use 99 for infinity):
0 3 4
3 0 99
4 99 0
Enter the source vertex:
0
Vertex   Distance       Path

0 -> 1          3               0 1
0 -> 2          4               0 2

D:\documents\College\SEM 5\Computer Network\Lab Programs>
```

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;
}
```

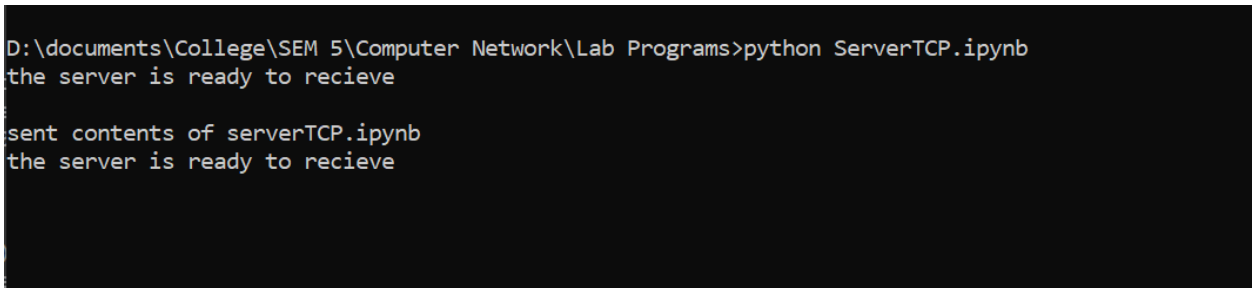


5. Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

ServerTCP.ipynb

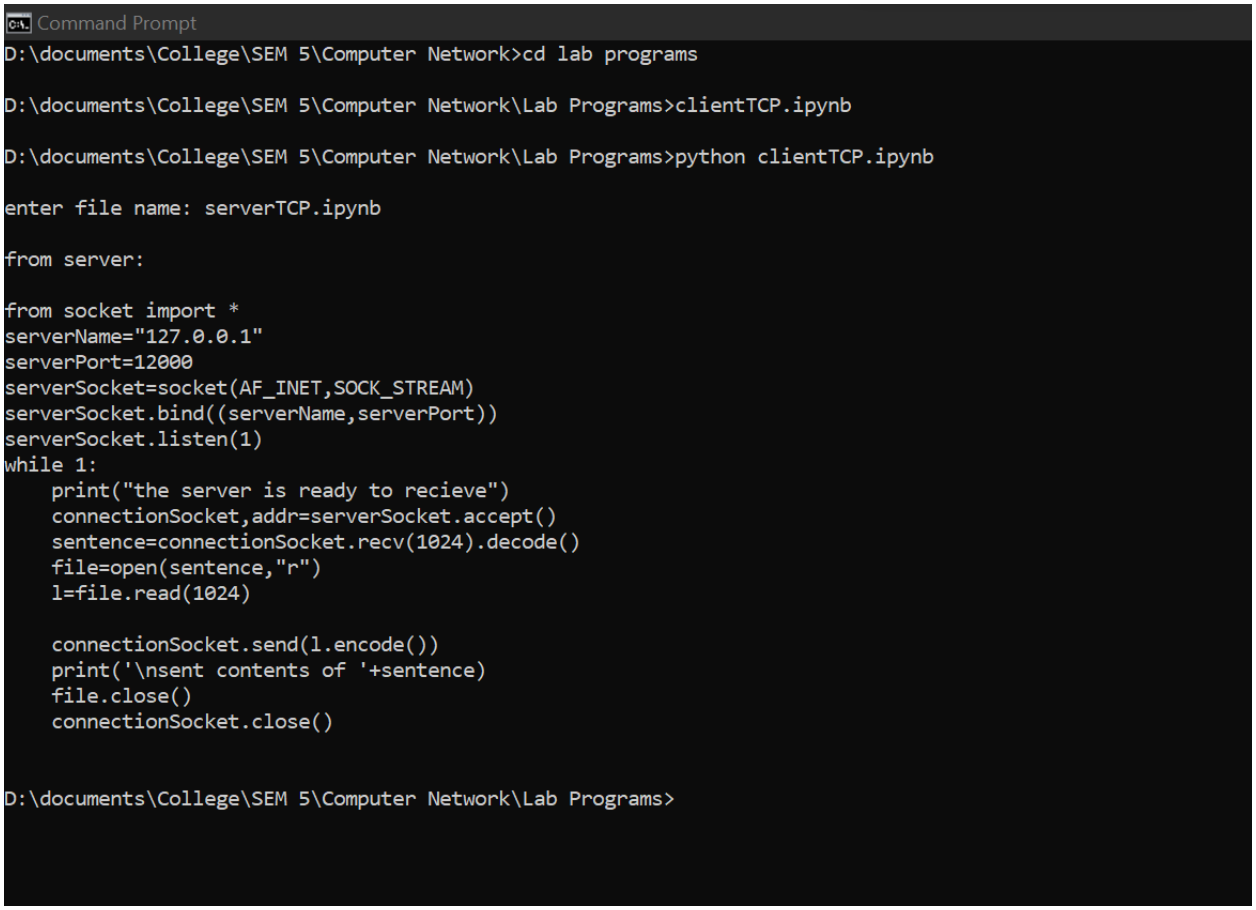
```
from socket import *
serverName="127.0.0.1"
serverPort=12000
```

```
serverSocket=socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
    print("the server is ready to recieve")
    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()
```



clientTCP.ipynb

```
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()
print('\nfrom server:\n')
print(filecontents)
clientSocket.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.

ServerUDP.ipynb

```
from socket import *
serverPort=12000
serverSocket=socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(("127.0.0.1",serverPort))
print("the server is ready to recieve")
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()
```

```
D:\documents\College\SEM 5\Computer Network\Lab Programs>python ServerUDP.ipynb
the server is ready to recieve

sent contents of serverUDP.ipynb
```

ClientUDP.ipynb

```
from socket import *
serverName="127.0.0.1"
serverPort=12000
clientSocket=socket(AF_INET,SOCK_DGRAM)
sentence=input("\nenter the 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()
```

```
D:\documents\College\SEM 5\Computer Network\Lab Programs>python ClientUDP.ipynb

enter the file name: serverUDP.ipynb

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 recieve")
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()
```