Aim:

To write a program to implement the data link layers framing methods such as

i) Character Stuffing ii) Bit Stuffing

Description:

Bit Stuffing:

It allows frames to contain arbitary number of bits and arbitary character size. The frames are seperated by seperating flag each frame begins and ends with a special bit pattern 01111110 called a flag byte when five consecutive set bits are encountered in the data. It automatically stuffs a '0' bit outgoing bit stream. This method finds its application in networks in which the change of data on physical medical contains some duplicated dilta.

Character Stuffing:

Character stuffing is also known as byte stuffing or character oriented framing and is same as that of bit stuffing actually operates on bytes. In bit stuffing special bit that is basically as ESC that has predefined pattren is generally added to data section of the data stream.

2.1) Write a Program to implement the data link layer framing method Character Stuffing. **Aim:** Program to implement Character Stuffing.

Program:

```
import java.util.*;
public class CharacterStuffing
{ public static void main(String[] args)
        {
           Scanner k =new Scanner (System.in);
     System.out.println("enter the string:\t");
     String s=k.nextLine();
     String str1; String
     str2=""; int i,m,j;
     m=s.length();
     System.out.println("original
     data:"+s); str1="dlestx";
     for(i=0;i<=m-1;i++)
     {
      if((s.charAt(i)=='d')&&(s.charAt(i+1)=='l')&&(s.charAt(i+2)=='e'))
     {
     str1=str1+"dle";
     }
     str1=str1+s.substring(i,i+1);
   }
   str1=str1+"dleetx";
   int p=str1.length();
   System.out.println("transmitted data:"+str1); for(i=6;i<p-
   6;i++)
   {

if((str1.charAt(i)=='d')&&(str1.charAt(i+1)=='l')&&(str1.charAt(i+2)=='e')&&(str1.charAt(i
+3)=='d')&&(str1.charAt(i+4)=='l')&&(str1.charAt(i+5)=='e'))
      { i=i+3;
      }
     str2=str2+str1.substring(i,i+1);
   }
   System.out.println("received data is:"+str2);
        }
}
```

Output:

```
enter the string:
Aditya dle c
original data:Aditya dle c
transmitted data:dlestxAditya dledle cdleetx
received data is:Aditya dle c
```

M. Vanu
18/8

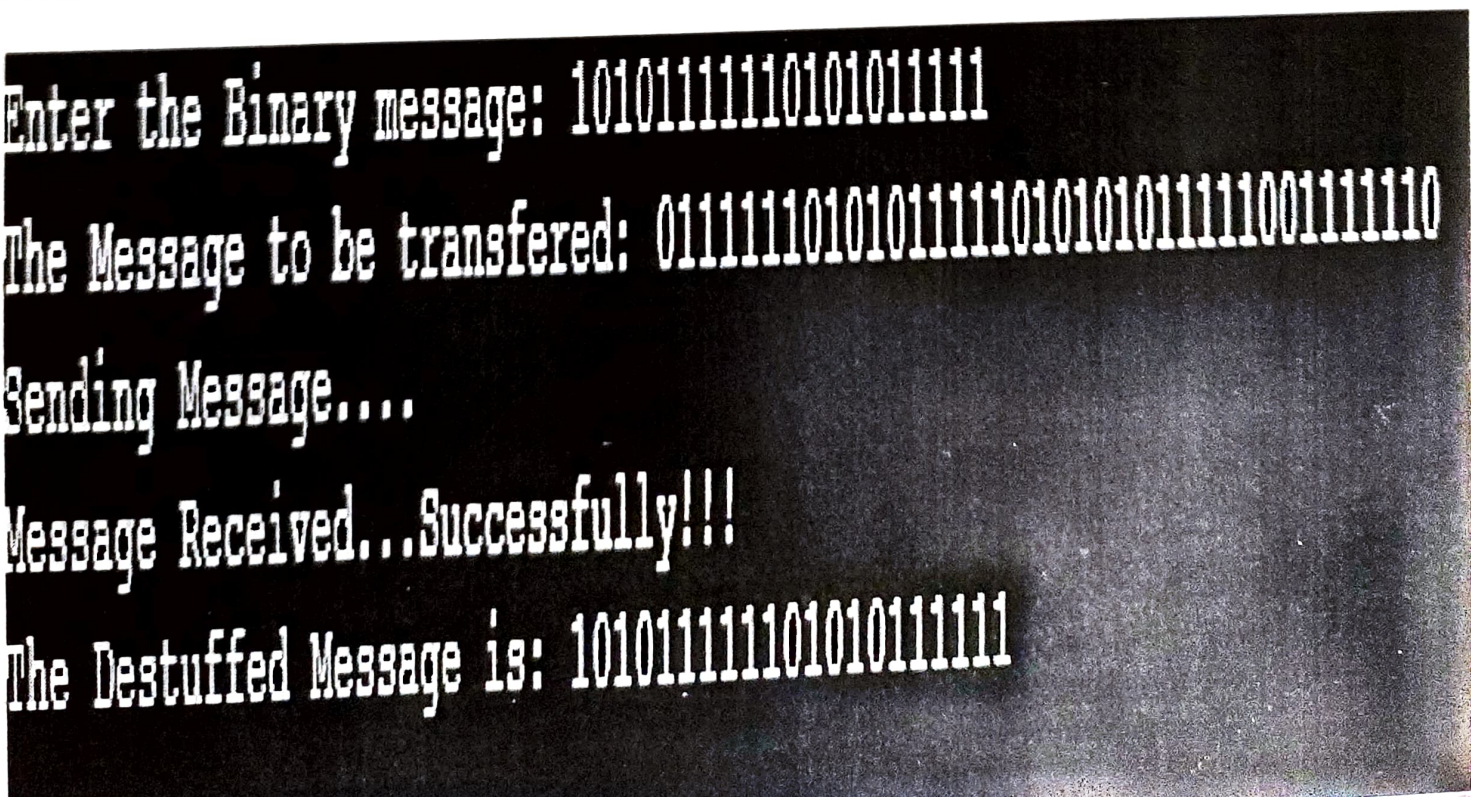
2.2) Write a Program to implement the data link layer framing method Bit Stuffing. **Aim:** Program to implement Bit Stuffing.

Program:

```
import java.util.*; public
class BitStuffing
{ public static void main(String[] args)
        {
          System.out.print("Enter the Binary message: ");
        Scanner sn=new Scanner(System.in);
        String data = sn.nextLine();
        String res = new String(); String
        out=new String();
        int counter = 0;
        for(int i=0;i<data.length();i++)
        {   if (data.charAt(i)!='1' && data.charAt(i)!='0')
                {
                   System.out.println("Enter only Binary values!!!");
                   return;
                } if(data.charAt(i)
            == '1')
                {
                   counter++;
                   res = res + data.charAt(i);
                }
            else
                {
                   res = res + data.charAt(i);
                   counter = 0;
                } if(counter
            == 5)
                { res = res + '0';
                   counter = 0;
                }
          }
        String inc=res;
        System.out.println("The Message to be transfered: " +inc);
        System.out.println("Sending Message ... ");
        counter=0;
        for(int i=0;i<res.length();i++)
        {

             if(res.charAt(i) == '1')
                {
```

```
            counter++; out = out +
            res.charAt(i);


        }
      else

        {
            out = out + res.charAt(i);
            counter = 0;
        } if(counter
      == 5)
        { if((i+2)!=res.length()) out =
            out + res.charAt(i+2);
            else out=out
            + '1'; i=i+2;
            counter = 1;
        }
}

 System.out.println("Message
Received...Successfully!!!"); System.out.println("The
Destuffed Message is: "+out); }
}
```

Output:



Aim:

To write a program to implement data link forming method.

Description:

Checksum is the error detection method used by upper layer protocols and is considered to be more reliable than LRC, VRC, etc.

This method makes the use of Checksum generates on Sender side and the checksum checker on recieverside.

Sender's End: The Sender adds the segments using its Complements arithmetic to get the sum. If the complement the sum to get the checksum and sends it along with data frames.

Reciever's End: The reciever's adds the incomming segments along with Checksum using is Complement arthemetic to get the sum and Complement it

If the result is zero, the recieved frames are accepted, otherwise they are discarded.

The advantage of checksum is detects all the error involving an odd number of bits as well as the errors involving an even number of bits.

Write a Program to implement data link layer farming method checksum.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
char a[20],b[20];
char sum[20],complement[20];
int i,length;
printf("Enter first binary string\n");
scanf("%s",&a);
printf("Enter second binary string\n");
scanf("%s",&b);
if(strlen(a)==strlen(b))
        {
        length = strlen(a);
        char carry='0';
        for(i=length-1;i>=0;i--)
{
        if(a[i]=='0' && b[i]=='0' && carry=='0') {
sum[i]='0';
carry='0';
}
else if(a[i]=='0' && b[i]=='0' && carry=='1')
{
sum[i]='1';
carry='0';
}
else if(a[i]=='0' && b[i]=='1' && carry=='0')
{
sum[i]='1';
carry='0';
}
else if(a[i]=='0' && b[i]=='1' && carry=='1')
{
sum[i]='0';
carry='1';
}
else if(a[i]=='1' && b[i]=='0' && carry=='0')
{
sum[i]='1';
carry='0';
```

```
}
else if(a[i]=='1' && b[i]=='0' && carry=='1') {
sum[i]='0';
        carry='1';

}
else if(a[i]=='1' && b[i]=='1' && carry=='0') {
sum[i]='0';
carry='1';
}
else if(a[i]=='1' && b[i]=='1' && carry=='1') {
sum[i]='1';
carry='1';
}
else
break;
}
printf("\nSum=%c%s",carry,sum);
for(i=0;i<length;i++) {
if(sum[i]=='0')
complement[i]='1';
else
complement[i]='0';
}
if(carry=='1')
carry='0';
else
carry='1';
printf("\nChecksum=%c%s",carry,complement);
        }
else {
                printf("\nWrong input strings");
        }}
```

Output:

```
Enter first binary string
101101
Enter second binary string
110010

Sum=1011111)
```

Aim:

To write a program for hamming code generation for error detection and correction

Description:

Hamming Code:

Hamming code is a linear code that is useful for error detection upto two immediate bit errors. It is capable of single bit errors.

In Hamming code, the source encodes the message by adding redundant bits in the message these redundant bits are mostly inserted and generated at certain positions in the message to accomplish error detection and correction process. We can encode and decode a message in hamming code.

Advantages:

Hamming code method is effective on networks where the data streams are given for the single bit errors.

Hamming code also helps you to intend bit containing error so that it can be corrected.

Disadvantages:

It can solve only one-bit errors.

EXPERIMENT-4

Write a program for Hamming Code generation for error detection and correction.

Program:

```
#include<stdio.h>
void main() {
        int data[10],dataatrec[10],c,c1,c2,c3,i;
        printf("Enter 4 bits of data one by one\n");
        scanf("%d%d%d%d",&data[0], &data[1], &data[2], &data[4]);
        data[6]=data[0]^data[2]^data[4];
        data[5]=data[0]^data[1]^data[4];
        data[3]=data[0]^data[1]^data[2];
        printf("The encoded data is \n");
        for(i=0;i<7;i++)
        printf("%d",data[i]);
        printf("\n\nEnter received data bits one by one\n");
        for(i=0;i<7;i++)
        scanf("%d",&dataatrec[i]);
        c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
        c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
        c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0]; c=c3*4+c2*2+c1;
        if(c==0) {
        printf("\nNo error while transmission of data\n");
        }
        else {
        printf("\nError on position %d",c);
        printf("\nData sent: ");
        for(i=0;i<7;i++)
        printf("%d",dataatrec[i]);
        printf("\nCorrect message is \n");
       if(dataatrec[7-c]==0)
       dataatrec[7-c]=1;
       else dataatrec[7-c]=0;
       for(i=0;i<7;i++) {
       printf("%d",dataatrec[i]);
}}}
```

```
Enter 4 bits of data one by one
1
0
1
0
The encoded data is
1010010

Enter received data bits one by one
1
0
1
0
1
1
0

Error on position 3
Data sent: 1010110
Correct message is
1010010
--------------------------------
Process exited after 19.51 seconds with return value 1
Press any key to continue . . .
```

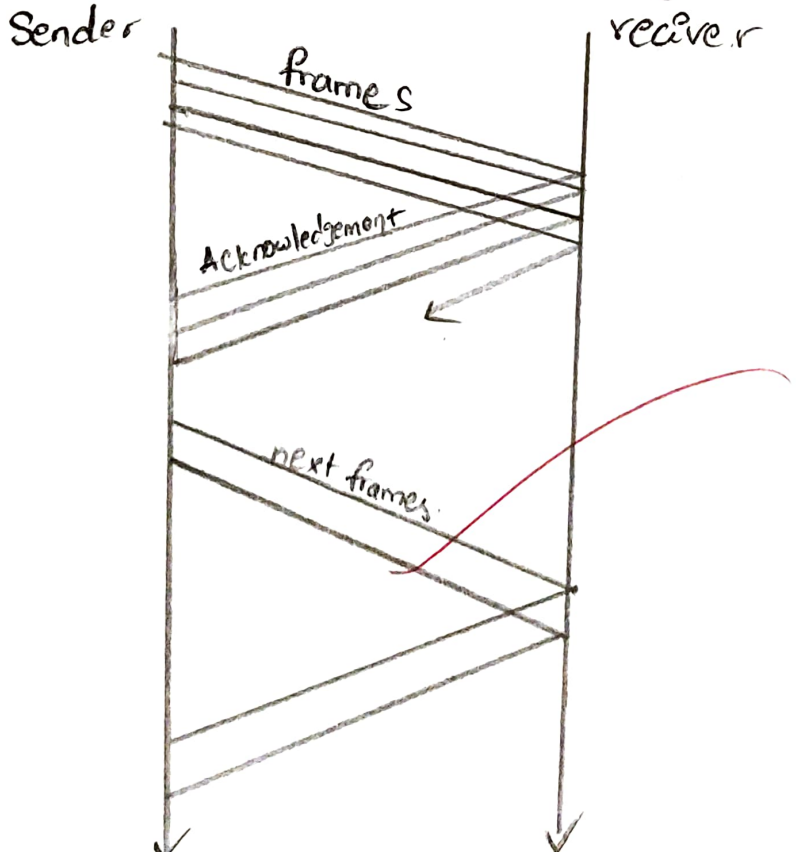
Aim:

To write a program to implement Sliding window protocol for GobackN.

Description:

Sliding window: In this flowcontrol mechanism, both sender and reciever aggree on the number of data frames after which the acknowledgement should send The term sliding window refers to imaginary boxes to send frames. In these protocols the sender has zero buffer Called the sliding window and reciever has a buffer called receving window.

Goback-N: In this protocol we are using and we can send sereval frames before reciering acknowledgements We keep a copy of these frames untill the acknowledgement recieve, if sits idle and do nothing



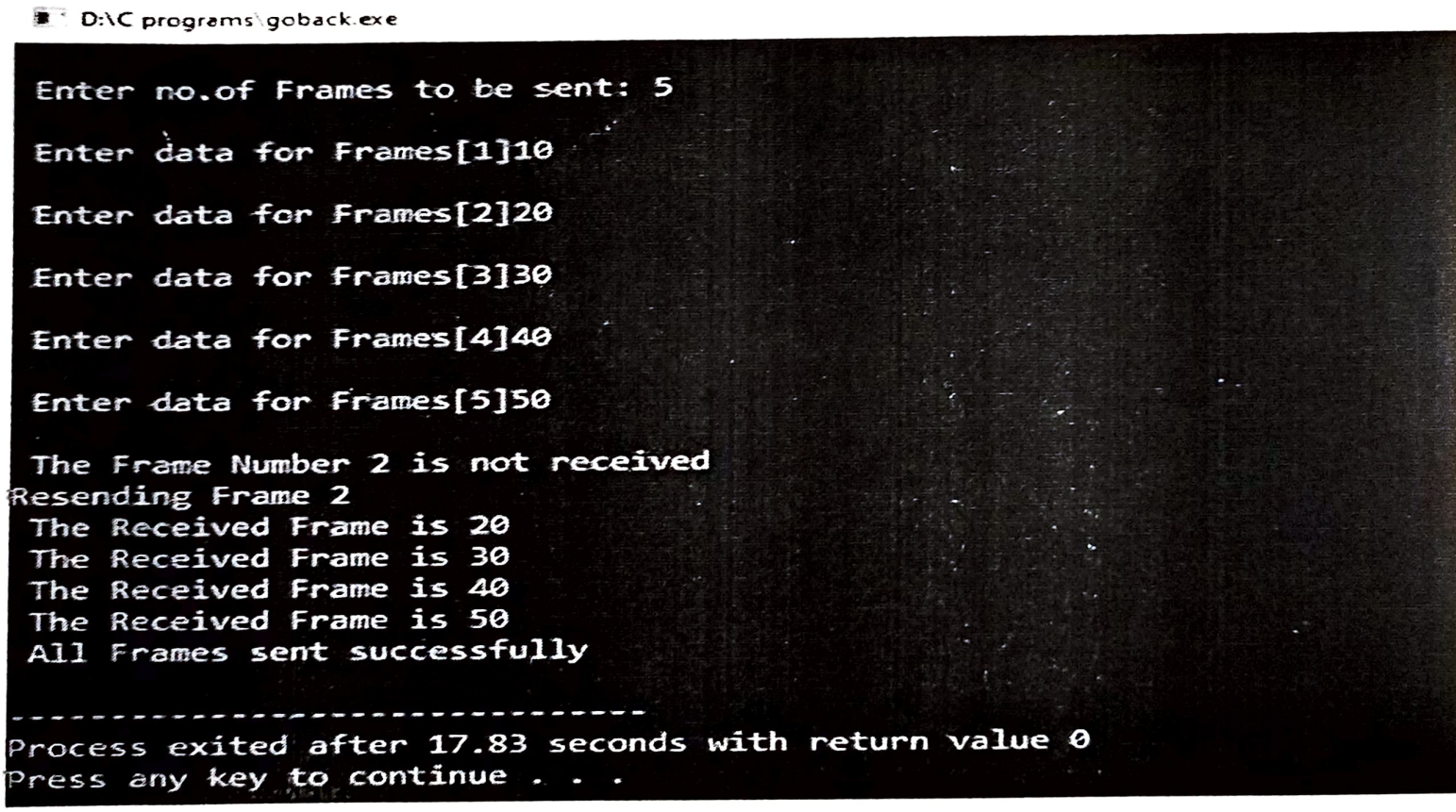
6)Write a Program to Implement Sliding window protocol for Goback N.

Program: #include<stdio.h> #include<stdlib.h> #include<math.h> #include<unistd.h>

```
int n,r;
struct frame
{
char ack; int data;
}frm[10];
int sender(void); void recvack(void);
void resend_gb(void); int main()
{
int c; sender(); recvack(); resend_gb();
printf("\n All Frames sent successfully\n");
}
int sender()
{
int i;
printf("\n Enter no.of Frames to be sent: "); scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\n Enter data for Frames[%d]",i); scanf("%d",&frm[i].data);
frm[i].ack='y';
}
return 0;
}
void recvack()
{
int i;
rand(); r=rand()%n; frm[r].ack='n'; for(i=1;i<=n;i++)

{

if(frm[i].ack=='n')

printf("\n The Frame Number %d is not received",r);

}

}

void resend_gb()

{
```

```
int i;
printf("\nResending Frame %d",r); for(i=r;i<=n;i++)
{
sleep(2); frm[i].ack='y';
printf("\n The Received Frame is %d",frm[i].data);
}
}
```

OUTPUT:



M. Vanu 15/9

Aim:

Write a program to implement sliding window protocol for Sllective repeat.

Description:

Selective repeat protocol is also called as selective repeat ARQ (Automatic Repeat Request), is a data link layer protocol that uses sliding window method for reliable delivery of data frames in which good frames are recieved and buffered.

Selective repeat provide protocol sending multiple frames depending upon the availability of frames on the sending window even if it does not recieve acknowledgement for any frame in the interim the maximum no. of frames that can be send depends upon the size of reciever window

The reciever records the sequence number of the earliest incorrect/unrecieved frame. It is continues sending the other frames.

7)Write a Program to implement Sliding window protocol for Selective repeat.

Program: #include<stdio.h> #include<stdlib.h> #include<math.h> #include<unistd.h>

```
int n,r;
struct frame
{
char ack; int data;
}frm[10];
int sender(void); void recvack(void); void resend_sr(void); int main()
{
int c; sender(); recvack(); resend_sr();
printf("\n All Frames sent successfully\n");
}
int sender()
{
int i;
printf("\n Enter no.of Frames to be sent: "); scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\n Enter data for Frames[%d]",i); scanf("%d",&frm[i].data);
frm[i].ack='y';
}
return 0;
}
void recvack()
{
int   i; rand(); r=rand()%n;
frm[r].ack='n';
{
if(frm[i].ack=='n')
printf("\n The Frame Number %d is not received",r);
}
}
void resend_sr()
{
printf("\nResending Frame %d",r); sleep(2);
frm[r].ack='y';
printf("\n The Received Frame is %d",frm[r].data);
}
```

```
Enter no.of Frames to be sent: 5

Enter data for Frames[1]10

Enter data for Frames[2]20

Enter data for Frames[3]30

Enter data for Frames[4]40

Enter data for Frames[5]50

The Frame Number 2 is not received
Resending Frame 2
The Received Frame is 20
All Frames sent successfully

--------------------------------
Process exited after 17.84 seconds with return value 0
Press any key to continue . . .
```

Aim:

To write a program to implement stop and wait protocol

Description:

It is the simplest flow control methods. In this method, the sender will transmit the one frame at a time to the reciever

This time is the sender's working time and the sender sends one data packet at a time sender sends the next packet only when it recieves the acknowledgement at the previous packet.

Therefore the idea of stop and wait protocol in the sender's side is very simple send one packet at a time and do not send another.

Reciver then consume the datapacket when the datapacket is consumed, reciever sends the acknowledgement.

EXPERIMENT-8

Write a Program to implement Stop and Wait Protocol.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<unistd.h>
int main()
{
int i,j,packet[30];
int fsize=(rand()%16)+1;
printf("\n\t Frame Size : %d \n",fsize);
printf("\n\t -------Data Log ------- \n");
printf("\n FRAME\t \tDATA\t WAITING\t \tACKNOW\t \tRESEND\t");
for(i=0;i<fsize;i++)
{
packet[i]=rand()%1000;
printf("\n %d \t\t %d",i+1,packet[i]);
while(j=0 || rand()%2==0)
{
sleep(1);
printf("\t1 ");
sleep(1);
for(j=2;rand()%2==0&&j<4;j++)
{
printf("%d ",j);
sleep(1);
}
if(j==4)
printf("\t\t\t NO \tRESENDING...\n %d \t %d",i+1,packet[i]);
else
break;
}
if(j==0)
{
sleep(1);
printf("\t0 ");
}
```

Exp No:
Date:

```
printf("\t\t\t YES \t\t NO");
}

printf("\n\n----------ALL DATA PACKETS SEND ----------\n");
return 0;
}
```

Output:

```
C:\Users\admin\Desktop\stop_and_wait.exe

        Frame Size : 10

        -------Data Log --------

FRAME           DATA      WAITING                 ACKNOW          RESEND
1                467      1 2                      YES              NO
2                724      1 2 3                    NO       RESENDING...
2       724      0                         YES                    NO
3                145      0                        YES              NO
4                627      0                        YES              NO
5                491      0                        YES              NO
6                942      0                        YES              NO
7                436      0                        YES              NO
8                604      1                        YES              NO
9                292      1                        YES              NO
10               716      1                        YES              NO

----------ALL DATA PACKETS SEND -----------

--------------------------------
Process exited after 19.29 seconds with return value 0
Press any key to continue . . .
```

Date : 13-10-2024

Aim:

To write a program for Congestion Control using leaky bucket algorithm.

Description:

Transport layer provides application multiplexing and demul-tiplexing, error detection, reliable data transfer, and Congestion Control services to the application layer. Application multiplexing and demultiplexing allow multiple applications to share and utilize both networks and system resources. Error detection and reliable data transformation respectively for application layer over the unreliable network links.

unlike the services, Congestion Control mechasim balances the network traffic as well as improve end to end Quality of services (QOS)

Therefore effective Congestion Control is as import-ant issue in transport layer.

Write a program for congestion

Program:

```
#include<stdio.h>
#include<stdlib.h>
struct packet{
int time;
int size;
}p[50];
int main(){
int i,n,m,k=0;
int bsize,bfilled,outrate;
printf("Enter the number of packets");
scanf("%d",&n);
printf("Enter packets in the order of their arrival time\n");
for(i=0;i<n;i++){
printf("Enter the time and size:");
scanf("%d%d",&p[i].time,&p[i].size);
}
printf("Enter the bucket size:");
scanf("%d",&bsize);
printf("Enter the output rate:");
scanf("%d",&outrate);
m=p[n-1].time;
i=1;
k=0;
bfilled=0;
while(i<=m || bfilled!=0){
printf("\n\nAt time %d",i);
if(p[k].time==i){
if(bsize>=bfilled + p[k].size){
bfilled=bfilled+p[k].size;
printf("\n%d bytes packet is inserted",p[k].size);
k=k+1;
}
else{
printf("\n%d bytes packet is discarded",p[k].size);
k=k+1;
```

```
}
}
if(bfilled==0){
printf("\nNo packets to transmit");

}
else if(bfilled>=outrate){
bfilled=bfilled-outrate;
printf("\n%d bytes transfered",bfilled);
bfilled=0;

}
printf("\nPackets in the bucket %d bytes",bfilled);

i++;

}}
```

Output

```
C:\Users\admin\Desktop\lecky.exe
Enter the number of packets6
Enter packets in the order of their arrival time
Enter the time and size:1 200
Enter the time and size:2 100
Enter the time and size:2 800
Enter the time and size:4 500
Enter the time and size:5 1000
Enter the time and size:6 200
Enter the bucket size:
700
Enter the output rate:100


At time 1
200 bytes packet is inserted
100 bytes transfered
Packets in the bucket 0 bytes

At time 2
100 bytes packet is inserted
0 bytes transfered
Packets in the bucket 0 bytes

At time 3
No packets to transmit
Packets in the bucket 0 bytes

At time 4
No packets to transmit
Packets in the bucket 0 bytes

At time 5
No packets to transmit
Packets in the bucket 0 bytes

At time 6
No packets to transmit
Packets in the bucket 0 bytes
```

Aim: To write a program to implement Dijkstra's algorithm to Compute the shortest path algorithm.

Description:

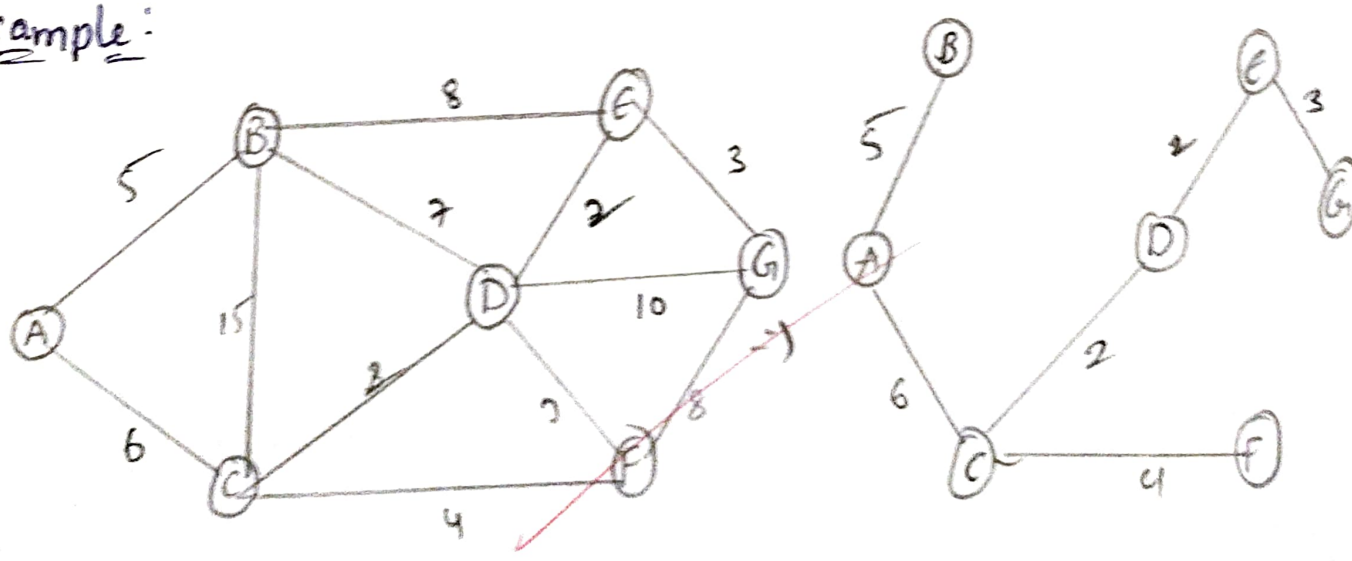
Dijkstra's algorithm finds the shortest path from a particular node called the source node to every other node in a connected graph.

It produces a shortest path tree from the source node as root node. It is profoundly used in Computer networks to generate optimal values with aim of minimizing routing costs.

Inputs → A graph representing the input

outputs → A shortest path tree, with s as root node.

Example:



EXPERIMENT-10

Write a Program to implement Dijkstra's algorithm to compute the Shortest path through a graph.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 30
#define LARGE 1000
struct state{
int len;
int pre;
int label;
};
struct state states[MAX];
int a[MAX][MAX];
int main(){
int i,j,s,d,n,tem,min,mini;
printf("Enter no.of vertices:");
scanf("%d",&n);
printf("\nEnter adjacency matrix\n");
/*Reading the adjacency matrix*/
for(i=0;i<n;i++)
for(j=0;j<n;j++){
printf("a[%d][%d]=",i,j);
scanf("%d",&a[i][j]);
}
/*Initially marking all nodes in the graph as temporary*/
for(i=0;i<n;i++){
states[i].len=LARGE;
states[i].label=0;
states[i].pre=-1;
}
printf("\nEnter source vertex:");
scanf("%d",&s);
printf("\nEnter destination vertex:");
scanf("%d",&d);
states[d].len=0;
states[d].label=1;
```

```
tem=d;
while(tem!=s){
for(i=0;i<n;i++){
if(a[tem][i]!=0 && states[tem].len+a[tem][i]<states[i].len &&
states[i].label==0){
states[i].len=states[tem].len+a[tem][i];
states[i].pre=tem;
}
}
min=LARGE;
mini=0;
/*Now find the vertex with the smallest length from all the nodes that are
temporary in the graph*/
for(i=0;i<n;i++){
if(states[i].len<LARGE && states[i].label==0){
min=states[i].len;
mini=i;
}
}
states[mini].label=1;
tem=mini;
}
tem=s;
printf("\nPath length:%d",states[s].len);
printf("\nPath\n");
printf("%d",tem);
do{
tem=states[tem].pre;
printf("%d",tem);
}while(tem!=d);
return 0;
}
```

```
enter no.of vertices:6

 enter adjacency matrix
a[0][0]=0
a[0][1]=7
a[0][2]=9
a[0][3]=0
a[0][4]=0
a[0][5]=14
a[1][0]=7
a[1][1]=0
a[1][2]=10
a[1][3]=15
a[1][4]=0
a[1][5]=0
a[2][0]=9
a[2][1]=10
a[2][2]=0
a[2][3]=11
a[2][4]=0
a[2][5]=2
a[3][0]=0
a[3][1]=0
a[3][2]=15
a[3][3]=11
a[3][4]=0
a[3][5]=6
a[4][0]=0
a[4][1]=0
a[4][2]=0
a[4][3]=0
a[4][4]=0
a[4][5]=9
a[5][0]=14
a[5][1]=0
a[5][2]=2
a[5][3]=0
a[5][4]=9
a[5][5]=0

 enter source vertex:0

 enter destination vertex:4

 path length:21
 path
0254
```

Aim:

To write a program to implement distance vector routing algorithm by obtaining routing table at each node.

Description:

The Distance vector algorithm is iterative, asynchoronous and distributed.

Distributed: It is distributed on that each node recieves information from one or more of its directly attached neighbours perform calculation and then distributes the result back its neighbours.

Iterative: It is iterative in that its process continues until no more information is available to be exchanged.

Asynchornous:

It doesnot require that all of its nodes operate in the lock step with each other

* The distance veetor algorithm is a dynamic algorithm
* It is mainly used on App or NET and RIP.
* Each router maintains a distance table known as vector

Week – 11

Write a Program to implement Distance vector routing algorithm by obtaining routing table at each node (Take an example subnet graph with weights indicating delay between nodes).

Program:

```
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{
  int costmat[20][20];
    int nodes,i,j,k,count=0;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes);
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)

        for(j=0;j<nodes;j++)
        {
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;                                                    {
            rt[i].dist[j]=costmat[i][j]; rt[i].from[j]=j;
        }
    }
        do
        {
            count=0; for(i=0;i<nodes;i++)
            for(j=0;j<nodes;j++)
            for(k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {
```

```
                rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                rt[i].from[j]=k;
                count++;
            }
        }while(count!=0);
        for(i=0;i<nodes;i++)
        {
            printf("\n\n For router %d\n",i+1);
            for(j=0;j<nodes;j++)
            {
                printf("\t\nnode %d via %d Distance %d
",j+1,rt[i].from[j]+1,rt[i].dist[j]);
            }
        }
    printf("\n\n");
}
```

Output:

```
Enter the number of nodes : 3

Enter the cost matrix :
0       2       7
2       0       1
7       1       0


 For router 1

node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 2 Distance 3

 For router 2

node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1

 For router 3

node 1 via 2 Distance 3
node 2 via 2 Distance 1
node 3 via 3 Distance 0


--------------------------------
Process exited after 32.26 seconds with return value 0
Press any key to continue . . .
```

M. Vani 27/11