EXPERIMENT-3

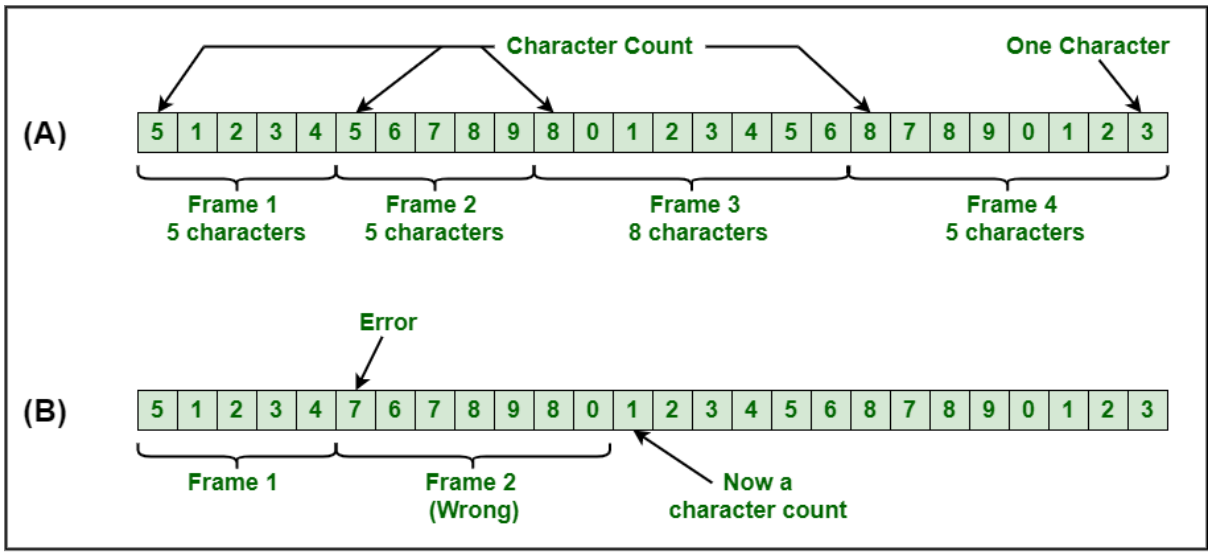
AIM: To implement the data link layer framing methods such as character count.

DESCRIPTION:

Character Count:

This method is rarely used and is generally required to count total number of characters that are present in frame. This is be done by using field in header. Character count method ensures data link layer at the receiver or destination about total number of characters that follow, and about where the frame ends.

There is disadvantage also of using this method i.e., if anyhow character count is disturbed or distorted by an error occurring during transmission, then destination or receiver might lose synchronization. The destination or receiver might also be not able to locate or identify beginning of next frame.



A Character Stream

(A) Without Errors
(B) With one Error

PROGRAM:

```
#include<stdio.h>

#include<string.h>

int main()

{

   char str[100];
```

```
int n,i,j,c=0,count=0;
printf("Enter the string:");
scanf("%s",str);
printf("Enter the number of frames:");
scanf("%d",&n);
int frames[n];
printf("Enter the frames size of the frames:\n");
for(i=0;i<n;i++)
{
   printf("Frame %d:",i);
   scanf("%d",&frames[i]);
}
printf("\n The number of frames : %d\n",n);
for(i=0;i<n;i++)
{
   printf("\n The content of frame %d\n:",i);
   j=0;
   while(c < strlen(str) && j < frames[i])
   {
      printf("%c",str[c]);
      if(str[c]!='\0')
      {
         count++;
      }
      c=c+1;
      j=j+1;
   }
```

```
        printf("\n Size of frame %d: %d\n\n",i,count);
        count=0;
    }
}
```

OUTPUT:

```
Enter the string:101011
Enter the number of frames:3
Enter the frames size of the frames:
Frame 0:3
Frame 1:4
Frame 2:5

 The number of frames: 3
 The content of frame 0 :101
 Size of frame 0: 3
 The content of frame 1:011
 Size of frame 1: 3
 The content of frame 2:
 Size of frame 2: 0
=== Code Execution Successful ===
```

RESULT:

The implementation of the data link layer framing methods as character count in c language have been executed successfully.

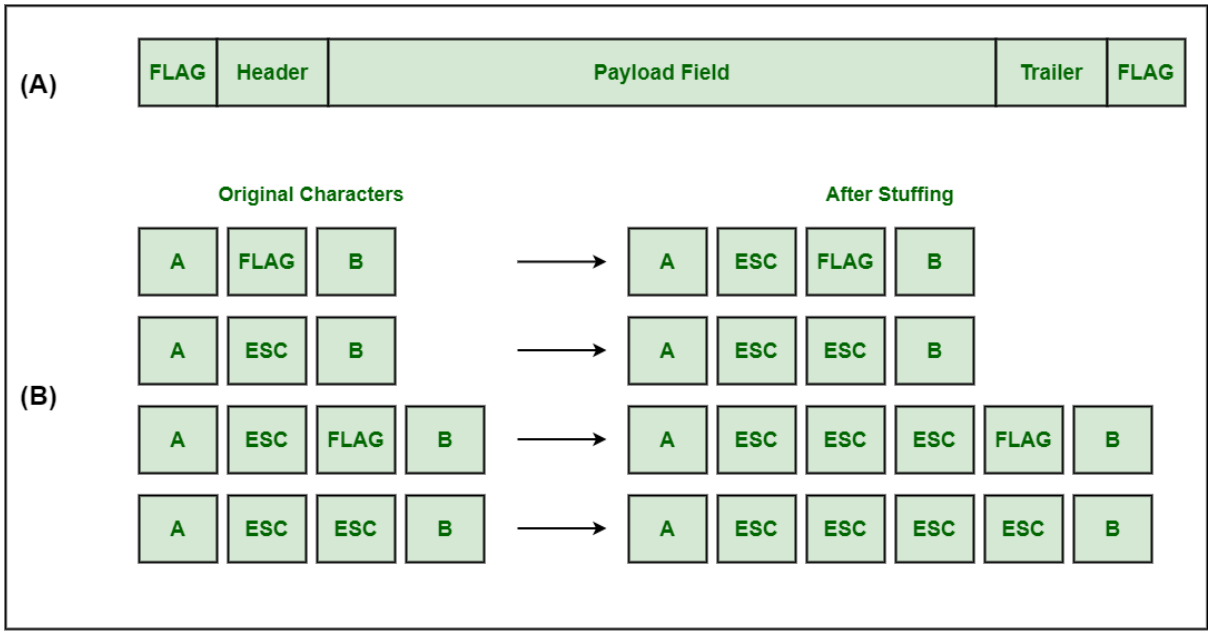
EXPERIMENT-4

AIM: To implement the data link layer framing method as character stuffing.

DESCRIPTION:

Character stuffing is also known as byte stuffing or character-oriented framing and is same as that of bit stuffing but byte stuffing actually operates on bytes whereas bit stuffing operates on bits. In byte stuffing, special byte that is basically known as ESC (Escape Character) that has predefined pattern is generally added to data section of the data stream or frame when there is message or character that has same pattern as that of flag byte.

But receiver removes this ESC and keeps data part that causes some problems or issues. In simple words, we can say that character stuffing is addition of 1 additional byte if there is presence of ESC or flag in text.



A Character Stuffing
(A) A frame delimited by flag bytes
(B) Four examples of byte sequences before and after byte stuffing

PROGRAM:

```
#include<stdio.h>

#include<string.h>

void main()

{
```

```
int i,k=0,n,j=6;
char s[100],res[100]="",a[100]="";
printf("Enter the string:");
gets(s);
strcpy(res,"dlestx");
for(i=0;s[i]!='\0';i++)
{
   if(s[i]=='d' && s[i+1]=='1' && s[i+2]=='e')
   {
      res[j]='d';
      res[j+1]='1';
      res[j+2]='e';
      j=j+3;
   }
   res[j]=s[i];
   j++;
}
strcat(res,"dleetx");
printf("stuffed char:%s",res);
n=strlen(res);
for(i=6;i<n-6;i++)
{
   if(res[i]=='d' && res[i+1]=='1' && res[i+2]=='e')
   {
     i=i+3;
   }
   a[k]=res[i];
```

```
    k++;
  }
  printf("\n Destuffed char:%s",a);
}
```

OUTPUT:

Enter the string: hello

Stuffed char: dlestxhellodleetx

Destuffed char: hello

RESULT:

The implementation of the data link layer framing method as character stuffing in c language have been executed successfully.

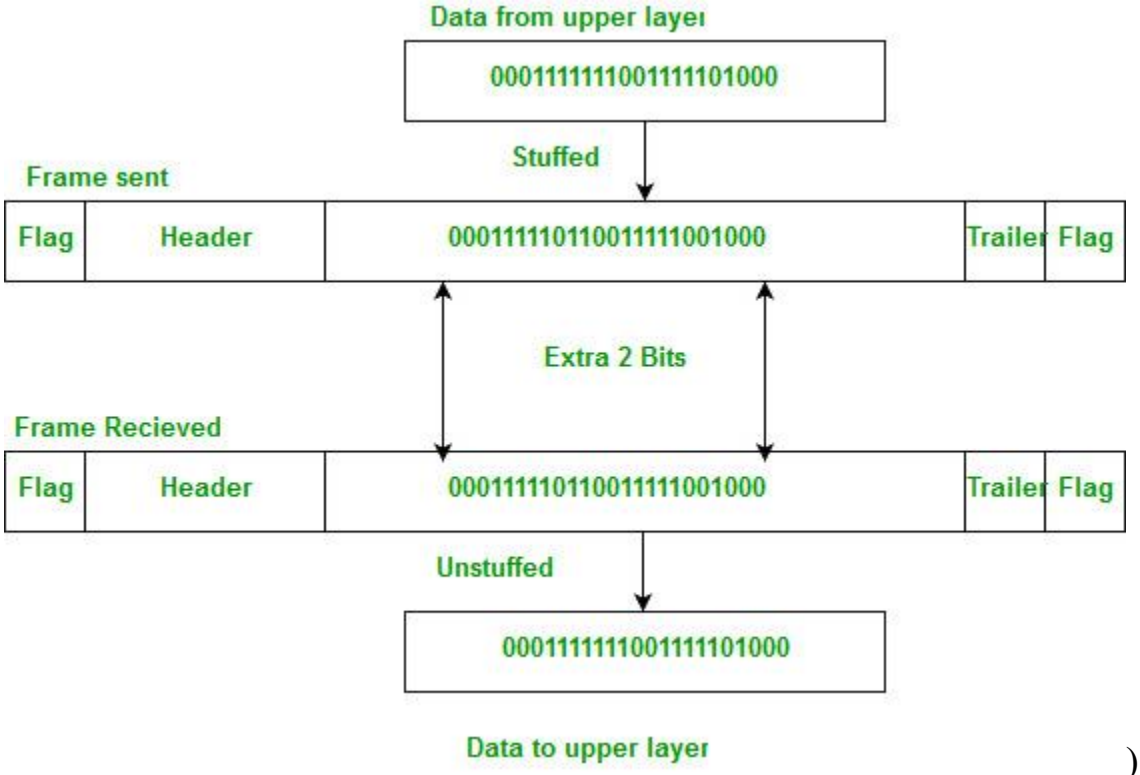
EXPERIMENT-5

AIM: To implement the data link layer framing method as bit stuffing.

DESCRIPTION:

Bit stuffing is the insertion of non information bits into data. Note that stuffed bits should not be confused with overhead bits.

Overhead bits are non-data bits that are necessary for transmission (usually as part of headers, checksums etc.



).

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int main() {
   int n,i,j=8,c=0,c1=0,k=8;
   char s[100], res[100];
   printf("enter string:");
   scanf("%s",s);
```

```
n=strlen(s);
strcpy(res,"01111110");
strcat(res," ");
for(i=0;i<n;i++)
{
   res[j]=s[i];
   j++;
   if(s[i]=='1')
   {
      c+=1;
      if(c==5)
      {
         res[j]='0';
         j++;
         c=0;
      }
   }
   else
   {
      c=0;
   }
}
strcat(res,"01111110");
printf("Stuffed string:");
printf("%s",res);
printf("\n Destuffed string:");
for(i=8;i < strlen(res)-8;i++)
```

```
    {
        if(res[i]=='1')
        {
            c1++;
        }
        else
        {
            c1=0;
        }
        printf("%c",res[i]);
        if(c1==5)
        {
            i++;
            c1=0;
        }
    }
    res[i]='\0';
}
```

OUTPUT:

enter string:011111101

Stuffed string:01111110011111010101111110

Destuffed string:011111101

RESULT: The implementation of the data link layer framing method as bit stuffing in c language have been executed successfully.

EXPERIMENT-6

AIM: To implement on a data set of characters the CRC polynomials CRC 12

DESCRIPTION:

Calculation of Polynomial Code (CRC) Checksum
1. For degree of generating polynomial G(x) = r, append r zero bits to low-order of frame. The frame now has m+r bits.
2. Divide the bit string corresponding to G(X) into the bit string xrM(x) mod(2)
3. Subtract the remainder R(x) from the bit string xrM(x) mod(2)

Frame: 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
G(X) = X4 + X + 1
Message after appending four 0's:
1 1 0 1 0 1 1 0 1 1 0 0 0 0
Remainder: 1110
Transmitted Frame:
1 1 0 1 0 1 1 0 1 1 1 1 1 0

PROGRAM:

```
#include<stdio.h>
 #include<conio.h>
 void main()
 {
    int f[150],g[150],r[150],t[150],i,j,k,m,n,c=0;
    clrscr();
    printf("\n Enter f: ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
       scanf("%d",&f[i]);
       t[i]=f[i];
```

```
}
printf("\n Enter g: ");
scanf("%d",&m);
for(i=0;i<m;i++)
{
   scanf("%d",&g[i]);
}
   for(i=n;i<n+m;i++)
   t[i]=0;
   for(i=0;i<n;i++)
   {
      k=i;
      if(t[i]==1)
      {
         for(j=0;j<m;j++,k++)
         {
            if(t[k]==g[j])
            t[k]=0;
            else
            t[k]=1;
         }
      }
   }
   printf("\n check sum is : ");
   for(i=n;i<m+n-1;i++)
   {
      printf(" %d ",t[i]);
```

```
        f[i]=t[i];

    }
    printf("\n\n BCS : ");
    for(i=0;i<m+n-1;i++)
    printf(" %d ",f[i]);
    printf("\n\nEnter received data : ");
    for(i=0;i<m+n-1;i++)
    scanf("%d",&r[i]);
    for(i=0;i<n;i++)
    {
       k=i;
       if(r[i]==1)
       {
          for(j=0;j<m;j++,k++)
          {
             if(r[k]==g[j])
             r[k]=0;
             else
             r[k]=1;
          }
       }
    }
printf(“Received data checksum\n”);
 for(i=n;i<m+n-1;i++)
 {
   printf(“%d”,r[i]);
```

```
  if(r[i]!=0)
  c=1;
 }
 if(c==1)
 printf("\n Error ");
 else
 printf("\n NO error");
 getch();
 }
```

OUTPUT:

Enter f: 24

111100001010101010000000

Enter g: 12

1 10000001 1 1 1

check sum is:010000001 10

BCS: 111100001010101010000000 01000000110

Enter received data: 1 1 1 100001010101010000000 0100 0000 1 10

Check sum is; 00000000000 NO error

RESULT:

The implementation of a data set of characters of CRC polynomials on CRC 12 in c language have been done successfully

EXPERIMENT-7

AIM: To implement Dijkstra‘s algorithm to compute the Shortest path through a graph.

DESCRIPTION:

Dijkstra's Algorithm is a Graph algorithm that finds the shortest path from a source vertex to all other vertices in the Graph (single source shortest path). It is a type of Greedy Algorithm that only works on Weighted Graphs having positive weights. The time complexity of Dijkstra's Algorithm is $O(V^2)$ with the help of the adjacency matrix representation of the graph. This time complexity can be reduced to O((V + E) log V) with the help of an adjacency list representation of the graph, where V is the number of vertices and E is the number of edges in the graph.

Dijkstra‘s algorithm:

- Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
- Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign the distance value as 0 for the source vertex so that it is picked first.
- While sptSet doesn’t include all vertices
    - Pick a vertex u that is not there in sptSet and has a minimum distance value.
    - Include u to sptSet.
    - Then update the distance value of all adjacent vertices of u .
        - To update the distance values, iterate through all adjacent vertices.
        - For every adjacent vertex v, if the sum of the distance value of u (from source) and weight of edge u-v , is less than the distance value of v , then update the distance value of v .

Note: We use a boolean array sptSet[] to represent the set of vertices included in SPT . If a value sptSet[v] is true, then vertex v is included in SPT , otherwise not. Array dist[] is used to store the shortest distance values of all vertices.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 10000
void main()
{
   int gptr[20][20],set[20],path[20],length[20],s,i,j,k,n,m,temp,c;

   clrscr();
   /* INPUT */
   printf("Enter No Of Vertexes\n");
   scanf("%d",&n);
   printf("Enter Adjacency Matrix of a Graph \n");
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
      scanf("%d",&gptr[i][j]);
    }
   printf("Enter Source node \n");
   scanf("%d",&s);
   /* INTIALIZATION PART */
   for(i=1;i<=n;i++)
   set[i]=0;
   for(i=1;i<=n;i++)
   {
      if(gptr[s][i]==0)
      {
```

```
            length[i]=INFINITY;
            path[i]=0;
        }
        else
        {
            length[i]=gptr[s][i];
            path[i]=s;
        }
    }
    set[s]=1;
    length[s]=0;
    /*ITERATION PART */
    temp=1;
    while(temp)
    {
        c=0;
        j=search_min(length,set,n);
        set[j]=1;
        for(i=1;i<=n;i++)
        {
            if(set[i]!=1)
            {
                if(gptr[i][j]!=0)
                {
                    if(length[j]+gptr[i][j]<length[i])
                    {
                        length[i]=length[j]+gptr[i][j];
```

```
                    path[i]=j;
                }
            }
        }
    }
for(i=1;i<=n;i++)
{
 if(set[i]==0)
 c++;
}
if(c==0)
 temp=0;
 else
 temp=1;
}
/*OUTPUT */
printf("length\tpath\n");
for(i=1;i<=n;i++)
printf("%d\t%d\n",length[i],path[i]);
printf("\n-----------------------------------------------------\n");
printf("\tPath\t\tLength\tShortest path \n");printf("----------------------\n");
printf("From(sourcevertex) To \n");
printf("-----------------------------------------------------\n");
printf("\t%d\n",s);
for(i=1;i<=n;i++)
{
   if(i!=s)
```

```
    printf("\t\t%4d\t%2d",i,length[i]);
    j=i;
    while(j!=s)
    {
       printf("\t%d->%d",j,path[j]);
       j=path[j];
    }
printf("\n");
 }
 getch();
 }
 /* search min function */
 search_min(int length[],int set[],int n)
 {
    int i,j,v,min=100;for(i=1;i<=n;i++)
    {
       if(set[i]==0)
       {
          if(length[i]<min)
          }
       }
    }
    return v;
 }
```

OUTPUT:

Enter No of Vertexes 5

Enter Adjacency Matrix of a Graph

0 1 6 0 0

1 0 2 3 5

6 2 0 4 2

0 3 4 0 2

0 5 2 2 0

Enter Source node

3

length path

3 2

2 3

0 0

4 3

2 3

Path Length Shortest path

From(sourcevertex) To

3

1 3 1->2 2->3

2 2 2->3

4 4 4->3

5 2 5->3

RESULT:

The implementation of Dijkstra‘s algorithm to compute the Shortest path through a graph in c language have been done successfully.

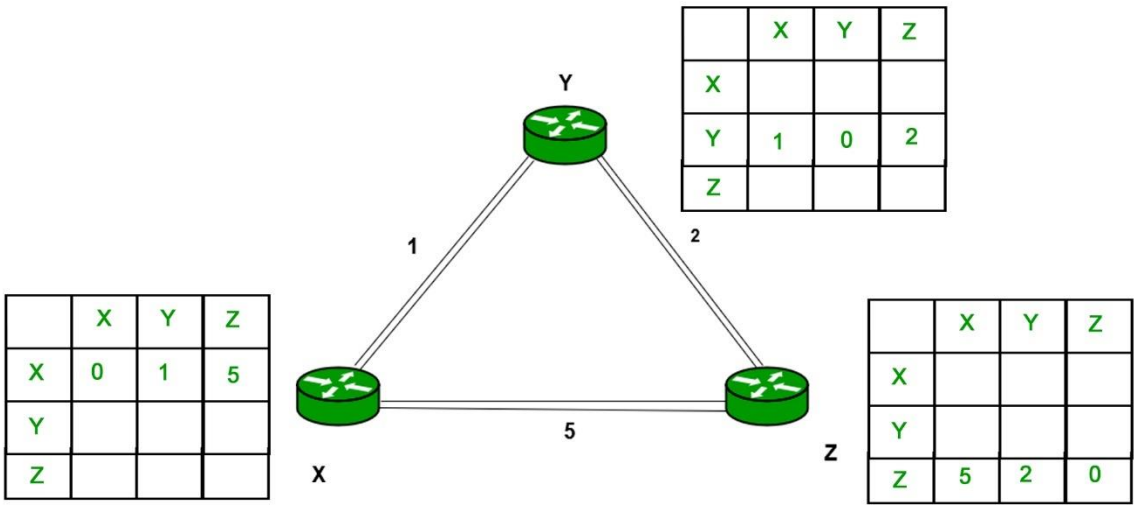
EXPERIMENT-8

AIM: Obtain hierarchical table by taking an example subnet graph with weights indicating delay between nodes.

DESCRIPTION:

Subnet Graph Example:

Consider a simple subnet graph with four routers: X, Y and Z. The edges between them represent the direct connections (adjacency) between routers, and the weights indicate the delay (in milliseconds) between these routers.



Graph Representation:

- Nodes (Routers): X, Y, Z
- Edges (Delays):
  - X-Y: 1 ms
  - X-Z: 5 ms
  - Y-Z: 2 ms

The adjacency matrix for this graph is as follows:

	X	Y	Z
X	0	1	3
Y	1	0	2
Z	5	2	0

Steps in the Experiment:

1. Input the Number of Routers and the Adjacency Matrix:
    - The user inputs the number of routers (n = 3).
    - The user inputs the adjacency matrix representing the graph.
2. Select the Router for Routing Table Construction:
    - The user selects a router, e.g., X, for which the routing table will be constructed.
3. Identify Neighbors and Input Delays:
    - The program identifies the neighbors of the selected router (X), which are Y and Z.
    - The user inputs the delay for direct connections from X to its neighbors.
4. Input Neighboring Routers' Routing Tables:
    - The user inputs the existing routing tables for the neighboring routers Y and Z.
5. Calculate the Minimum Delay:
    - For each router, the program calculates the minimum delay using the delays and the routing tables provided.
    - The result is stored in a hierarchical table that represents the best path from the selected router to each other router.
6. Display the Final Routing Table:
    - The program outputs the hierarchical routing table for the selected router.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
struct full
{
    char line[10],dest[10];
    int hops;
}f[20];
 main()
{
int nv,min,minver,i;
char sv[2],temp;
clrscr();
printf("\nEnter number of vertices:");
scanf("%d",&nv);
printf("\n Enter source vertex: ");
scanf("%s",sv);
printf("\n Enter full table for source vertex %s :\n",sv);
for(i=0;i<nv;i++)
scanf("%s %s %d",f[i].dest,f[i].line,&f[i].hops);
printf("\n HIERARCHIAL TABLE\n\n");
for(i=0;i<nv;)
{
if(sv[0]==f[i].dest[0])
{
printf("\n %s %s %d",f[i].dest,f[i].line,f[i].hops);
i++;
```

```
}
else
{
min=1000;
minver=0;
temp=f[i].dest[0];
while(temp==f[i].dest[0])
{
if(min>f[i].hops)
{
min=f[i].hops;
minver=i;
}
i++;
}
printf("\n %c %s %d ",temp,f[minver].line,f[minver].hops);
}
}
getch();
}
```

INPUT/OUTPUT:

```
 Enter number of vertices: 8
Enter source vertex :1A
Enter full table for source vertex 1A :
1A - -
1B 1B 1
1C 1C 1
```

2A 1B 1

2B 1B 2

3A 1C 2

3B 1C 3

4A 1C 3

HIERARCHIAL TABLE

1A – 0

1B 1B 1

1C 1C 1

2 1B 1

3 1C 2

4 1C 3

RESULT:

The program of obtaining a hierarchical table by taking an example subnet graph with weights indicating delay between nodes in c language have been executed successfully.

EXPERIMENT-9

AIM: obtain Routing table for each node using distance vector routing algorithm

ALGORITHM:

Input: Graph and a given vertex src

Output: Shortest distance to all vertices from src. If there is a negative weight cycle, then shortest distances are not calculated, negative weight cycle is reported.

1) This step initializes distances from source to all vertices as infinite and distance to source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.
2) This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph.
   …..a) Do following for each edge v-u
   ……If dist[v] > dist[u] + weight of edge uv, then update dist[v]
   …….dist[v] = dist[u] + weight of edge uv
3) This step reports if there is a negative weight cycle in graph. Do following for each edge u-v
   ……If dist[v] > dist[u] + weight of edge uv, then "Graph contains negative weight cycle" The idea of step 3 is, step 2 guarantees shortest distances if graph doesn't contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle

PROGRAM:

```
#include<stdio.h>

#include<conio.h>

#include<math.h>

main()

{

int i,j,k,nv,sn,noadj,edel[20],tdel[20][20],min;
```

```
char sv,adver[20],ch;
clrscr();
printf("\n ENTER THE NO.OF VERTECES:");
scanf("%d",&nv);
printf("\n ENTER THE SOURCE VERTEX NUM,BER AND NAME:");
scanf("%d",&sn);
flushall();
sv=getchar();
printf("\n NETER NO.OF ADJ VERTECES TO VERTEX %c",sv);
scanf("%d",&noadj);
for(i=0;i(tdel[j][i]+edel[j]))
{
 min=tdel[j][i]+edel[j];
ch=adver[j];
}
if(i!=sn-1)
printf("\n%d %c",min,ch);
else
printf("\n0 -");
}
getch();
}
```

INPUT/OUTPUT:

```
ENTER THE NO.OF VERTECES:12
ENTER THE SOURCE VERTEX NUMBER AND NAME:10 J
ENTER NO.OF ADJ VERTECES TO VERTEX 4
ENTER TIME DELAY and NODE NAME:8 A
ENTER TIME DELAY and NODE NAME:10 I
```

ENTER TIME DELAY and NODE NAME:12 H

ENTER TIME DELAY and NODE NAME:6 K

ENTER THE TIME DELAY FROM A to ALL OTHER NODES:

0 12 25 40 14 23 18 17 21 9 24 29

ENTER THE TIME DELAY FROM I to ALL OTHER NODES:

24 36 18 27 7 20 31 20 0 11 22 33

ENTER THE TIME DELAY FROM H to ALL OTHER NODES:

20 31 19 8 30 19 6 0 14 7 22 9

ENTER THE TIME DELAY FROM K to ALL OTHER NODES:

21 28 36 24 22 40 31 19 22 10 0 9

DELAY VIA--VERTEX

8 a

20 a

28 i

20 h

17 i

30 i

18 h

12 h

10 i

0 –

6 k

15 k

RESULT:

The program of obtaining Routing table for each node using distance vector routing algorithm in c language have been executed successfully.

EXPERIMENT-10

AIM: Find minimum code and minimum spanning tree for a given subnet of hosts.

DESCRIPTION:

This technique is widely used because it is simple and easy to understand. The idea of this algorithm is to build a graph of the subnet with each node of the graph representing a router and each arc of the graph representing a communication line. To choose a route between a given pair of routers. The algorithm just finds the boat cast between them on the graph.

PROGRAM:

// Write a 'c' program for broadcast tree from subnet of host

```
#include<stdio.h>

int p,q,u,v,n;

int min=99, mincost=0;

int t[50][2],i,j;

int parent[50],edge[50][50];

main()

{

Clrscr();

printf("\n Enter the number of nodes");

scanf("%d",&n);

for(i=0;i<n;i++)

{

Printf("%c\t",65+i);

Parent[i]=-1
```

```
}
printf(“\n”);
for(i=0;i<n;i++)
{
Printf(“%c”,65+i);
for(j=0;j<n;j++)
scanf(“%d”,&edge[i][j]);
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
if(edge[i][j]!=99)
if(min>edge[i][j])
{
min=edge[i][j]
u=i;
v=j;
}
p=find(u);
q=find(v);
if(p!=q)
{
t[i][0]=u;
t[i][1]=v;
mincost=mincost+edge[u][v];
sunion(p,q);
}
```

```
else
t[i][0]=-1;
t[i][1]=-1;
}
min=99;
}
Printf(“Minimum cost is%d\n Minimum spanning tree is\n”,mincost);
for(i=0;i<n;i++)
if(t[i][0]!=-1 && t[i][1]!=-1)
{
printf(“%c%c%d”,65+t[i][0],65+t[i][1],edge[t[i][0]] [t[i][1]]
printf(“\n”);
}
getch();
}
Sunion(int I,int m)
{
parent[]=m;
}
find(int I )
{
if(parent[I]>0)
I=parent[I];
return I;
}
```

OUTPUT:

Enter the number of nodes 4

A B C D

A1 3 5 6

B6 7 8 9

C2 3 5 6

D1 2 3 7

Minimun cost is 9

Minimum spanning tree is

B A 6

C A 2

D A 1

RESULT:

The program of finding minimum code and minimum spanning tree for a given subnet of hosts in c language have been executed successfully