Computer Networks

List of Experiments

PROGRAM - 1

1. Write a program to implement:

a. Read the dotted decimal IP address and print in the binary format.

b. Find the classification of an IP address.

1 a) Read the dotted decimal IP address and prints in the binary format.

Algorithm:

Step1: First divide the number by given number.
Step 2: The remainder will be either 0 or 1.
Step 3: Write down the remainder.
Step 4: Divide the remaining number without the remainder
Step 5: Again, the remainder will be either 1 or 0.
Step 6: Write down the remainder to the left of the previous remainder.
Step 7: Repeat this until you end up with 0.

Source code:

```
#include<stdio.h>
#include<math.h>
void iptobin(char[15],int[4][8]);
void main()
{
        char mip[15];
        int bmip[4][8],b[32];
        int i,j,k;
        for(i=0;i<4;i++)
        for(j=0;j<8;j++)
        bmip[i][j]=0;
        printf("Enter machine IP Address\n");
        scanf("%s",mip);
        iptobin(mip,bmip);
        k=0;
        for(i=0;i<4;i++)
        for(j=7;j>=0;j--)
        b[k++]=bmip[i][j];
        printf("MACHINE IP in binary notation \n");
        for(k=0;k<32;k++)
        printf("%d",b[k]);
        printf("\n");
}
void iptobin(char mip[15],int ip[4][8])
{
        int i=0,j=-1,k,num1,num[4];
        char split[4][8];
```

```
        i=0;
        while(mip[i]!='\0'){
                j++;k=0;
                while(mip[i]!='.'&& mip[i]!='\0'){
                 split[j][k]=mip[i];
                 k++;i++;
                }
                split[j][k]='\0';
                i++;
        }
        for(i=0;i<4;i++){
                printf(" %s",split[i]);
                printf("\n"); }
        for(i=0;i<4;i++)
                num[i]=atoi(split[i]);
        for(j=0;j<4;j++){
                num1=num[j];
                for(i=0;num1!=0;num1/=2,i++)
                ip[j][i]=num1%2;
        }
}
```

Output:

Enter Machine IP address:
128.255.0.0
The equivalent binary is:
10000000111111110000000000000000

1 b) Find the classification of an IP address.

Algorithm:

Step1: start
Step 2: initialize the variables.
Step 3: enter the IP address
Step 4: call function bin
Step 5: classification of class
Step 6: end

Source code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>

char addr[16],bnr[32];
```

```
char* bin();
char* cobin(char[]);

main()
{
        int ch;
        char *res;
        int i,count=0;
        printf("enter the Address\n");
        scanf("%s",addr);
        res=bin();
        for(i=0;i<4;i++)
         {
                if(res[i]=='1')
                count++;
                else
                break;
         }

        if(count==0)
                printf("the ip is class A\n");
        else if(count==1)
                printf("the ip is class B\n");
        else if(count==2)
                printf("the ip is class C\n");
        else if(count==3)
                printf("the ip is class D\n");
        else if(count==4)
                printf("the ip is class E\n");
}

char* bin()
{
          char  *t,*temp,*res,*r;
         int i,j,k;
         res=(char*)malloc(32);
         t=(char*)malloc(4);
         r=(char*)malloc(8);
         temp=(char*)malloc(16);
         strcpy(temp,addr);
         for(i=0;i<4;i=i+1)
        {
                t=strtok(temp, ".");
                r=cobin(t);
                strcat(res,r);
                temp=temp+strlen(t)+1;
```

```
        }
        return res;
}

char* cobin(char *s)
{
         int  i=7;
        char *a;
        int temp,j;
        temp=atoi(s);
        j=temp;
        a=(char*)malloc(8);
        while(temp>=1)
        {
                if(temp%2==0)
                a[i]='0';
                else
                a[i]='1';
                i=i-1;
                temp=temp/2;
      }
        for(j=i;j>=0;j--)
                 a[j]='0';
        return a;
}
```

Output:

```
Enter the Address
250.222.21.124
The IP is class E
```

PROGRAM - 2

2. Write a program to implement:

a. Read the binary format IP address and print in the dotted decimal format

b. Find the network id, host id of a given IP address.

2 a) Read the binary format IP address and print in the dotted decimal format.

Algorithm:

Step 1: First divide the number by
Step 2: The remainder will be either 0 or 1.
Step 3: Write down the remainder.
Step 4: Divide the remaining number without the remainder by
Step 5: Again, the remainder will be either 1 or 0.
Step 6: Write down the remainder to the left of the previous remainder.
Step 7: Repeat this until you end up with 0.

Source code:

```
int power(int x,int y)
{
        int i,sum=1;
        for(i=1;i<=y;i++)       sum*=x;
        return sum;
}

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include  "pow.h"
char addr[16],bnr[32];
void codec();
int dec(char[]);
main()
{
        char *res;
        printf("conversion to decimal\n");
        printf("enter the 32 bit binary Adress\n");
      scanf("%s",bnr);
      codec();
}

void codec()
{
        int i,j,k=0;
        char res[8];
```

```
        for(i=0;i<32;i=i+8)
        {
                for(j=i;j<i+8;j++)
                res[k++]=bnr[j];
                k=dec(res);
                printf("%d.",k);
                k=0;
        }
}
int dec(char *s)
{
        int i,j=0,sum=0;
        for(i=7;i>=0;i--)
                {
                        if(s[i]=='1')
                           sum=sum+power(2,j);
                    j++;
                }
        return sum;
}
```

Output:

```
conversion to decimal
enter the 32 bit binary Adress
10100000101000001010000010100000
160.160.160.160
```

2 b) Find the network id, host id and the subnet id of a given IP address

Algorithm:

Step 1: To find the first valid IP address, copy the subnet number and add 1 to the fourth octet
Step2: To find the last valid IP address, copy the broadcast address and subtract 1 to the fourth octet

Source code:

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

int readIPAddress(int ipaddr[])
 {

        char decstring[4][4];
        int i;
        printf("Enter 32-bit IP address in dotted decimal
```

```
                notation (xxx.xxx.xxx.xxx): ");
              fflush(stdout);
        if (scanf("%[^.].%[^.].%[^.].%[^\r\n]", decstring[0], decstring[1],decstring[2],
decstring[3]) < 4)
                {
                        printf("Invalid input string (wrong IP address
                        format)\n");
                        return 0;
                }
                /* convert each string to integer*/
                 for (i=0; i<4; i++)
                 {
                        ipaddr[i] = atoi(decstring[i]);
                        /* atoi converts a string to an integer */
                         if (ipaddr[i] < 0 || ipaddr[i] > 255)
                       {
                                printf("Invalid input string
                                (incorrect numbers for IP address)\n");
                                return 0;
                          }
                    }
                                return 1;
                }

        void showNetworkHost(int ipaddr[], int numNetwork)
        {
                 int i;
                 printf("Network portion is: ");
                 for (i=0; i<=numNetwork; i++)
                          printf("%d.", ipaddr[i]);
                 printf("\n");
                 printf("Host portion is: ");
                for (i=numNetwork+1; i<4; i++)
                        printf(".%d", ipaddr[i]);
                 printf("\n");
        }

        int main()
        {
                 int choice;
                 char bin[4][8];
                 char bin2[4][9];
                 int ipaddr[4];
                 int mask[4];
                 int slash;
                 char temp[80];
```

```
        do
        {
                printf("\nSelect from one of the choices:\n\n");
                printf("1) display the network and host portions
                separately\n");
                printf("2) Quit the program\n");
                printf("\nEnter your choice: ");
                fflush(stdout);
                scanf("%d", &choice);
                scanf("%[^\n]", temp);
                scanf("%c", temp);

                switch (choice)
                {
                 case 1 :
                     /* Convert a dotted decimal ip form to its
                     class */
                     if (!readIPAddress(ipaddr))
                                break;
                      if (ipaddr[0] < 128){
                                showNetworkHost(ipaddr, 0);
                      }
                     else if (ipaddr[0] < 192){
                                showNetworkHost(ipaddr, 1);
                     }
                     else if (ipaddr[0] < 224){
                                showNetworkHost(ipaddr, 2);
                      }
                     else if (ipaddr[0] < 240){
                                showNetworkHost(ipaddr, 3);
                     }
                     else     printf("Invalid address\n");
                    break;

                 case 2 :
                                system("clear"); //clear program
                           exit(0); // close program

                 default:
                                printf("I don't know the option %d.\n", choice);
                                printf("Try again.\n");
                           break;
                }
        } while(1);
}
```

Output:

Select from one of the choices:

1) Display the network and host portions separately
2) Quit the program
Enter your choice: 1
Enter 32-bit IP address in dotted decimal notation (xxx.xxx.xxx.xxx):
10.10.10.10
Network portion is: 10.
Host portion is: .10.10.10

Select from one of the choices:
1) Display the network and host portions separately
2) Quit the program
Enter your choice: 2

PROGRAM - 3

3. Write a program to implement:

a. The Error – Detection Technique: Cyclic – Redundancy - Check.

b. Farming Methods: Bit stuffing & Character Stuffing.

3 a) The Error-Detection Technique: cyclic redundancy check

Algorithm:

Begin
Step 1: Declare I,j,fr[8],dupfr[11],recfr[11],tlen,flag,gen[4],genl,frl, rem[4] as integer
Step 2: initialize frl=8 and genl=4
Step 3: initialize i=0
Step 4: Repeat step(5to7) until i<frl
Step 5: read fr[i]
Step 6: dupfr[i]=fr[i]
Step 7: increment i
Step 8: initialize i=0
Step 9: repeat step(10to11) until i<genl
Step 10: read gen[i]
Step 11: increment i
Step 12: tlen=frl+genl-1
Step 13: initialize i=frl
Step 14: Repeat step(15to16) until i<tlen
Step 15: dupfr[i]=0
Step 16: increment i
Step 17: call the function remainder(dupfr)
Step 18: initialize i=0
Step 19: repeat step(20 to 21) until j<genl
Step 20: recfr[i]=rem[j]
Step 21: increment I and j

Step 22: call the function remainder(dupfr)
Step 23: initialize flag=0 and i=0
Step 24: Repeat step(25to28) until i<4
Step 25: if rem[i]!=0 then
Step 26: increment flag
Step 27: end if
Step 28: increment i
Step 29: if flag=0 then
Step 25: print frame received correctly
Step 25: else
Step 25: print the received frame is wrong
End
Function: Remainder(int fr[])
Begin
Step 1: Declare k,k1,I,j as integer
Step 2: initialize k=0;
Step 3: repeat step(4 to 14) until k< frl
Step 4: if ((fr[k] == 1) then
Step 5: k1=k
Step 6: initialize i=0, j=k
Step 7: repeat step(8 to 9) until i< genl
Step 8: rem[i] =fr[j] exponential gen[i]
Step 9: increment I and j
Step 10: initialize I = 0
Step 11: repeat step(12to13) until I <genl
Step 12: fr[k1] = rem[i]
Step 13: increment k1 and i
Step 14: end if
End

Source code:

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

void main()
{
        char f[25]=" ",cal[25]=" ",g[20]="  ",r[20]=" ";
        int m,i,j,l,k,x,degree;
        clrscr();
        printf("enter original frame\n");
        gets(f);
        m=strlen(f);  /*to find length of original frame*/
        printf("\n enter generator polynomial\n");
        gets(g);
        l=strlen(g);        /* to find length of generator polynomial */
        for(i=1;i<l;i++)
                strcat(f,"0");
        strcat(f,"\0");
        puts(f);
```

```
  for(i=1;i<l;i++)
          cal[i]=f[i];
  cal[i]='\0';
  x=l;
  printf("the length of f is %d\n",strlen(f));

  while(x<=strlen(f))     /* To calculate check sum */
  {
          for(i=1;i<l;i++)
                  if(cal[i]==g[i])
                  r[i-1]='0';
                  else
                          if(cal[i]!=g[i])
                                  r[i-1]='1';
                          if(x==strlen(f))
                          {
                                  r[l-1]='\0';
                                  break;
                           }
                                  r[l-1]=f[x];
                          k=0;
                          while(r[k]=='0')
                                  {
                                          for(i=0;i<l-1;i++)
                                          r[i]=r[i+1];
                                          x++;
                                          r[l-1]=f[x];
                                  }
                          strcat(r,'\0');
                          strcpy(cal,r);
                          x++;
  }
  printf("the checksum is %s\n",r);        /* printing check sum */
  for(i=m,j=0;r[j]!='\0';i++,j++)
          f[i]=r[j];
  printf("finally the frame i.e, to be sent is %s,\length=%d",f,strlen(f));
for(i=1;i<l;i++)
          cal[i]=f[i];
  x=l;
  while(x<=strlen(f))
  {
          for(i=1;i<l;i++)
          if(cal[i]==g[i])
                  r[i-1]='0';
                  else
                  if(cal[i]!=g[i])
                  r[i-1]='1';
  if(x==strlen(f))
  {
            r[l-1]='\0';
```

```
                break;
            }
            r[l-1]=f[x];
            k=0;
            while(r[k]=='0')
            {
                for(i=0;i<l-1;i++)
                   r[i]=r[i+1];
              x++;
              r[l-1]=f[x];
            }
            strcat(r,'\0');
            strcpy(cal,r);
            x++;
            }
            /* printing received data */
            printf("the recived data check sum is %s\n",r);
            getch();
    }
```

Output:

```
enter original frame
100100
enter generator polynomial
1101
100100000
the length of f is 9
the checksum is 001
finally the frame i.e, to be sent is 100100001,
length=9the recived data check sum is 000
```

3 b) Framing Methods: Bit stuffing and character stuffing

Bit Stuffing:

Algorithm:

Begin
Step 1: Read frame length n
Step 2: Repeat step (3 to 4) until i<n(: Read values in to the input frame (0's and1's)
Step 3: initialize I i=0;
Step 4: read a[i] and increment i
Step 5: Initialize i=0, j=0,count =0
Step 6: repeat step (7 to 22) until i<n
Step 7: If a[i] == 1 then
Step 8: b[j] = a[i]
Step 9: Repeat step (10 to 18) until (a[k] =1 and k<n and count <5)
Step 10: Initialize k=i+1;
Step 11: Increment j and b[j]= a[k];
Step 12: Increment count;
Step 13: if count =5 then

Step 14: increment j,
Step 15: b[j] =0
Step 16: end if
Step 17: i=k;
Step 18: increment k
Step 19: else
Step 20: b[j] = a[i]
Step 21: end if
Step 22: increment I and j
Step 23: print the frame after bit stuffing
Step 24: repeat step (25 to 26) until i< j
Step 25: print b[i]
Step 26: increment i
END

Source code:

```
#include<stdio.h>
void bitstuff();
char msg[30],st[30],dst[30],flag[30],enc[30],dec[30];
int q,count=0,t,i,j,elen,count1;
main()
{
  printf("enter the message \n");
  scanf("%s",msg);
  printf("Enter the flag \n");
  scanf("%s",flag);
  q=strlen(flag);
   bitstuff();
 }
        void bitstuff()
        {
              for(i=0,j=0;msg[i]!='\0';i++,j++)
              {
                    st[j]=msg[i];
                    if(st[j]=='1')
                    count=count+1;
                    else
                    count=0;
                    if(count==(q-3))
                    {
                    st[j+1]='0';
                    count=0;
                    j++;
                    }
              }
              st[j]='\0';
              printf("The message after stuffing is %s \n",st);
              strcpy(enc,flag);
              strcpy(dec,st);
```

```
        strcat(enc,dec);
        strcat(enc,flag);
        printf("\n the stuffed message along with flag is : %s \n",enc);
        elen=strlen(enc);
        for(i=q,j=0;i<(elen-q);i++,j++)
        dst[j]=enc[i];
        dst[j]='\0';
        count1=0;
        printf("the message is %s \n",dst);
  for(i=0,j=0;dst[i]!='\0';i++,j++){
                dec[j]=dst[i];
                if(dec[j]=='1')
                count1=count1+1;
                else
                count1=0;
                if(count1==(q-3) && dst[i+1]=='0'){
                i++; count1=0;
                }
        }
        dec[j]='\0';
        printf("The destuffed message is : %s \n",dec);
}
```

Output:

enter the message
anil
Enter the flag
100
The message after stuffing is v0e0n0u0g0o0p0a0l0
the stuffed message along with flag is : 100v0e0n0u0g0o0p0a0l0100
the message is v0e0n0u0g0o0p0a0l0
The destuffed message is: anil

Character Stuffing:

Algorithm:

Begin
Step 1: Initialize I and j as 0
Step 2: Declare n and pos as integer and a[20],b[50],ch as character
Step 3: read the string a
Step 4: find the length of the string n, i.e n-strlen(a)
Step 5: read the position, pos
Step 6: if pos > n then
Step 7: print invalid position and read again the position, pos
Step 8: end if
Step 9: read the character, ch
Step 10: Initialize the array b , b[0…5] as ’d’, ’l’, ’e’, ’s’ ,’t’ ,’x’ respectively
Step 11: j=6;

Step 12: Repeat step[(13to22) until i<n
Step 13: if i==pos-1 then
Step 14: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’, ‘e’ ,’ch, ’d’, ‘l’,‘e’ Respectively
Step 15: increment j by 7, i.e j=j+7
Step 16: end if
Step 17: if a[i]==’d’ and a[i+1]==’l’ and a[i+2]==’e’ then
Step 18: initialize array b, b[13…15]=’d’, ‘l’, ‘e’ respectively
Step 19: increment j by 3, i.e j=j+3
Step 20: end if
Step 21: b[j]=a[i]
Step 22: increment I and j;
Step 23: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’,‘e’ ,’e’,‘t’, ‘x’,‘\0’ respectively
Step 24: print frame after stiuffing
Step 25: print b
End

Source code:

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
    int i,j,k;
    char data[50],temp[50],etx[50]="DLEETX";
    char stx[50]="DLESTX",dl[50]="DLE";
    clrscr();
    cout<<"Enter data to be transmitted"<<endl;
    gets(data);
    for(i=0;i<strlen(data);i++)          /* Character stuffing process */
      if(data[i]=='D'&&data[i+1]=='L'&&data[i+2]=='E')
     {
            strcpy(temp,data+i);
            data[i]='\0';
        strcat(data,dl);
        strcat(data,temp);
            i+=3;
     }
    strcat(data,etx);
    strcpy(temp,stx);
    strcat(temp,data);
    strcpy(data,temp);
    puts("Data after stuffing") ;
    printf("\n");
    puts(data);
  k=strlen(stx);
  strcpy(temp,data+k);
```

```
  j=strlen(temp)-strlen(etx);
  strcpy(data,temp);
  data[j]='\0';
  for(i=0;i<strlen(data);i++)              /* Character destuffing process */
if(data[i]=='D'&&data[i+1]=='L'&&data[i+2]=='E')
{
                        strcpy(temp,data+i+3);
                        data[i]='\0';
                        strcat(data,temp);
}
printf("\nData after destuffing\n ",);
puts(data);
getch();
}
```

Output:

```
Enter data to be transmitted
anil
Data after stuffing
DLESTXanilDLEETX
Data after destuffing
anil
```

4. Write a program to implement the following Static - Routing algorithms:

a. Shortest – Path Routing (Using Dijkstra's).

b. Multicast Routing (Using Minimum Spanning Tree algorithms).

4. Write a program to implement the following Static - Routing algorithms:

a. Shortest – Path Routing (Using Dijkstra's).

b. Multicast Routing (Using Minimum Spanning Tree algorithms).

4a) Write a program in "C" to implement shortest path routing (using Dijkstra's) Algorithms.

Algorithm:

Begin
Step1: Declare array path [5] [5], min, a [5][5], index, t[5];
Step2: Declare and initialize st=1,ed=5
Step 3: Declare variables i, j, stp, p, edp
Step 4: print "enter the cost "
Step 5: i=1
Step 6: Repeat step (7 to 11) until (i<=5)
Step 7: j=1
Step 8: repeat step (9 to 10) until (j<=5)
Step 9: Read a[i] [j]
Step 10: increment j
Step 11: increment i
Step 12: print "Enter the path"
Step 13: read p
Step 14: print "Enter possible paths"
Step 15: i=1
Step 16: repeat step(17 to 21) until (i<=p)
Step 17: j=1
Step 18: repeat step(19 to 20) until (i<=5)
Step 19: read path[i][j]
Step 20: increment j
Step 21: increment i
Step 22: j=1
Step 23: repeat step(24 to 34) until(i<=p)
Step 24: t[i]=0
Step 25: stp=st
Step 26: j=1
Step 27: repeat step(26 to 34) until(j<=5)
Step 28: edp=path[i][j+1]
Step 29: t[i]= [ti]+a[stp][edp]
Step 30: if (edp==ed) then
Step 31: break;
Step 32: else
Step 33: stp=edp
Step 34: end if
Step 35: min=t[st]
Step 36: index=st
Step 37: repeat step(38 to 41) until (i<=p)

Step 38: min>t[i]
Step 39: min=t[i]
Step 40: index=i
Step 41: end if
Step 42: print" minimum cost" min
Step 43: print" minimum cost pth"
Step 44: repeat step(45 to 48) until (i<=5)
Step 45: print path[index][i]
Step 46: if(path[idex][i]==ed) then
Step 47: break
Step 48: end if
End

Source Code:

```
#include<stdio.h>
#define MAX 80
#define INF 9999
int dist[MAX][MAX],d[MAX],visit[MAX],n;
void init()
{
        int i;
        for(i=1;i<=n;i++)
        {
        d[i]=INF;
        visit[i]=0;
        }
}
void dijkst(int s)
{

        int i,k,mini;
        d[s]=0;
        for(k=1;k<=n;k++){
        mini=-1;
        for(i=1;i<=n;i++)
        if(!visit[i] && ((mini==-1)||(d[i]<d[mini])))
        mini=i;
        visit[mini]=1;
        for(i=1;i<=n;i++)
        if(dist[mini][i])
        if(d[mini]+dist[mini][i]<d[i])
        d[i]=d[mini]+dist[mini][i];
        }
        printf("The distance matrix is:\n");
        for(i=1;i<=n;i++)
        printf("%d...%d-->%d\n",s,i,d[i]);
        printf("\n");
}
main()
{
```

```
        int i,j,s;
        system("clear");
        printf("Enter the number of nodes:");
        scanf("%d",&n);
        printf("Enter the %d X %d matrix(0 if no path):\n",n,n);
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        scanf("%d",&dist[i][j]);
        init();
        printf("Enter the source to start:");
        scanf("%d",&s);
       dijkst(s);
}
```

Output:

```
Enter the number of nodes:4
Enter the 4 X 4 matrix(0 if no path):
2 4 3 1
1 3 2 4
4 2 1 2
3 4 3 1
Enter the source to start:1
The distance matrix is:
1...1-->0
1...2-->4
1...3-->3
1...4-->1
```

4b) Write a c Program Multicast Routing (Using Minimum Spanning Tree algorithms).

Algorithm:

Begin
Step1: Declare array path [5] [5], min, a [5][5], index, t[5];
Step2: Declare and initialize st=1,ed=5
Step 3: Declare variables i, j, stp, p, edp
Step 4: print “enter the cost “
Step 5: i=1
Step 6: Repeat step (7 to 11) until (i<=5)
Step 7: j=1
Step 8: repeat step (9 to 10) until (j<=5)
Step 9: Read a[i] [j]
Step 10: increment j
Step 11: increment i
Step 12: print “Enter the path”
Step 13: read p
Step 14: print “Enter possible paths”
Step 15: i=1
Step 16: repeat step(17 to 21) until (i<=p)
Step 17: j=1
Step 18: repeat step(19 to 20) until (i<=5)
Step 19: read path[i][j]
Step 20: increment j
Step 21: increment i
Step 22: j=1
Step 23: repeat step(24 to 34) until(i<=p)
Step 24: t[i]=0
Step 25: stp=st
Step 26: j=1
Step 27: repeat step(26 to 34) until(j<=5)
Step 28: edp=path[i][j+1]
Step 29: t[i]= [ti]+a[stp][edp]
Step 30: if (edp==ed) then
Step 31: break;
Step 32: else
Step 33: stp=edp
Step 34: end if
Step 35: min=t[st]
Step 36: index=st
Step 37: repeat step(38 to 41) until (i<=p)
Step 38: min>t[i]
Step 39: min=t[i]
Step 40: index=i
Step 41: end if
Step 42: print” minimum cost” min
Step 43: print” minimum cost pth”
Step 44: repeat step(45 to 48) until (i<=5)
Step 45: print path[index][i]
Step 46: if(path[idex][i]==ed) then

Step 47: break
Step 48: end if
End

Source code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define maxVertices 1000
#define maxEdges 1000000
int graph[maxVertices][maxVertices];
/* Input graph must be undirected,weighted and connected*/
typedef struct Edge
{
	int from,to,weight;
}Edge;
int compare(const void * x,const void * y)
{
	return (*(Edge *)x).weight - (*(Edge *)y).weight;
}
Edge E[maxEdges];
int parent[maxVertices];
void init(int vertices)
{
	int iter=0;
	for(iter=0;iter<vertices;iter++)
	{
		parent[iter]=-1;
	}

}
int Find(int vertex)
{
	if(parent[vertex]==-1)return vertex;
	return parent[vertex] = Find(parent[vertex]); /* Finding its parent as well as updating the
parent of all vertices along this path */
}
int Union(int parent1,int parent2)
{
	/* This can be implemented in many other ways. This is one of them */
	parent[parent1] = parent2;
}
void Kruskal(int vertices,int edges)
{
	memset(graph,-1,sizeof(graph)); /* -1 represents the absence of edge between them */
	/* Sort the edges according to the weight */
	qsort(E,edges,sizeof(Edge),compare);
	/* Initialize parents of all vertices to be -1.*/
	init(vertices);
```

```
	int totalEdges = 0,edgePos=0,from,to,weight;
	Edge now;
	while(totalEdges < vertices -1)
	{
		if(edgePos==edges)
		{
			/* Input Graph is not connected*/
			exit(0);
		}
		now = E[edgePos++];
		from = now.from;
		to = now.to;
		weight=now.weight;
		/* See If vetices from,to are connected. If they are connected do not add this edge. */
		int parent1 = Find(from);
		int parent2 = Find(to);
		if(parent1!=parent2)
		{
			graph[from][to] = weight;
			Union(parent1,parent2);
			totalEdges++;
		}
	}

}
int main()
{
	int vertices,edges;
printf("enter no of vertices and edges");
	scanf("%d%d",&vertices,&edges);
	int iter,jter;
	int from,to,weight;
	for(iter=0;iter<edges;iter++)
	{
printf("enter starting nodeand ending node and weight");
scanf("%d%d%d",&from,&to,&weight);
		E[iter].from = from;
		E[iter].to = to;
		E[iter].weight = weight;
	}
	/* Finding MST */
	Kruskal(vertices,edges);
	/* Printing the MST */
	for(iter=0;iter<=vertices;iter++)
	{
		for(jter=0;jter<=vertices;jter++)
		{
			if(graph[iter][jter]!=-1)
			{
				printf("Vertex %d and %d, weight %d\n",iter,jter,graph[iter][jter]);
```

```
                }
            }
        }
        return 0;
}
```

Output:

```
Enter the no of vertices and edges
2
2
Enter stating node and endig node
1 1 3
1 2 2
2 1 4
2 2 6
1—3 is 8
```

PROGRAM - 5

5. Write a program to implement the following Dynamic - Routing algorithm: Distance - Vector Routing (Using Fulkerson – Ford or Bell man Ford)

```
#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);//Enter the 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];//initialise the distance equal to cost matrix
         rt[i].from[j]=j;
      }
   }
      do
      {
         count=0;
         for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from
the node i to k using the cost matrix
         //and add the distance from k to node j
         for(j=0;j<nodes;j++)
         for(k=0;k<nodes;k++)
            if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
            {//We calculate the minimum distance
               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 :
2 3 4
2 5 4
4 3 5
For router 1
node 1 via 1 Distance 0
node 2 via 2 Distance 3
node 3 via 3 Distance 4
For router 2
node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 4
For router 3
node 1 via 1 Distance 4
node 2 via 2 Distance 3
node 3 via 3 Distance 0

PROGRAM - 6

6. Write a program for congestion control using Leaky bucket algorithm

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 byte packet is inserted",p[k].size);
                    k=k+1;
                   }
      else
              {    printf("\n%d byte packet is discarded",p[k].size);
                   k=k+1;
               }
   }
```

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

       printf("\nPackets in the bucket %d byte",bfilled);       i++;
    }   return 0;
}
```

Example Output1 :

```
Enter the number of packets: 3

Enter packets in the order of
they are arrival time

Enter the time and size: 1 100
Enter the time and size: 2 400
Enter the time and size: 3 600

Enter the bucket size: 500
Enter the output rate:  200

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

At time 2
400 byte packet is inserted
200 bytes transfered
Packets in the bucket 200 byte

At time 3
600 byte packet is discarded
200 bytes transfered
Packets in the bucket 0 byte
```

Example Output 2

```
Enter the time and size: 1 100
Enter the time and size: 3 200
Enter the time and size: 2 400
Enter the time and size: 4 600
Enter the bucket size: 200
Enter the output rate: 100

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

At time 2
No packets to transmitted
Packets in the bucket 0 byte

At time 3
200 byte packet is inserted
100 bytes transfered
Packets in the bucket 100 byte

At time 4
100 bytes transfered
Packets in the bucket 0 byte
```

PROGRAM - 7

Write a program for simple RSA algorithm to encrypt and decrypt the data? (7)

Program:

```
#include <stdio.h> #include <string.h> #include <conio.h> #include <math.h>
int mult(unsigned int x, unsigned int y, unsigned int n)
{
unsigned long int k=1;
int j;
for (j=1; j<=y; j++)
k = (k * x) % n;
return (unsigned int) k;
}
void main ()
{
char msg[100];
unsigned int pt[100], ct[100], n, d, e, p, q, i;
printf("ENTER MESSAGE : ");
gets(msg);
//strcpy(pt, msg);
for(i=0;i<strlen(msg);i++)
pt[i]=msg[i];
n = 253; d = 17; e = 13;
printf("\nCT = ");
for(i=0; i<strlen(msg); i++)
ct[i] = mult(pt[i], e,n);
for(i=0; i<strlen(msg); i++)
printf("%d ", ct[i]);
printf("\nPT = ");
for(i=0; i<strlen(msg); i++)
printf("%c", pt[i]);
for(i=0; i<strlen(msg); i++)
pt[i] = mult(ct[i], d,n) ;
}
```

OUTPUT:

ENTER MESSAGE: COMPUTER SCIENCE AND INFROMATION TECHNOLOGY

CT = 111 107 110 247 72 189 115 169 219 172 111 200 115 12 111 115 219 76 12 206 219 200 12 185 169 107 110 76 189 200 107 12 219 189 115 111 216 12 107 43 107 4 221

PT = COMPUTER SCIENCE AND INFROMATION TECHNOLOGY