Program – 6

Design, develop & execute a program to implement error detection using the Cyclic Redundancy Check – Consultative Committee for International Telegraphy and Telephony (CRC-CCITT) – (16-bit) algorithm.

C Code: (P6.c)

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)

char t[128],cs[128],g[]="10001000000100001";
int a,e,c;
void xor()
{
   for(c=1;c<N;c++)
      cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc()
{
   for(e=0;e<N;e++)
      cs[e]=t[e];
   do
   {
      if(cs[0]=='1')
         xor();
      for(c=0;c<N-1;c++)
         cs[c]=cs[c+1];
      cs[c]=t[e++];
   }while(e<=a+N-1);
}
int main()
{
   printf("\nEnter poly : ");
   scanf("%s",t);
   printf("\nGenerating Polynomial is : %s",g);
   a=strlen(t);
   for(e=a;e<a+N-1;e++) t[e]='0';
      printf("\nModified t[u] is :  %s",t);
   crc();
```

```
    printf("\nChecksum is : %s",cs);
    for (e=a;e<a+N-1;e++) t[e]=cs[e-a];
       printf("\nFinalCodeword is : %s",t);
    printf("\nTest Error detection 0(yes) 1(no) ? : ");
    scanf("%d",&e);
    if (e==0)
    {
       printf("Enter position where error is to inserted : ");
       scanf("%d",&e);
       t[e]=(t[e]=='0')?'1':'0';
       printf("Errorneous data: %s\n",t);
    }
    crc();
    for(e=0;(e<N-1)&&(cs[e]!='1');e++);
       if(e<N-1)
          printf("Error detected.");
       else
          printf("No Error Detected.");
    return 0;
}
```

Output:

```
Enter poly: 1011101
Generating Polynomial is: 10001000000100001
Modified t[u] is:  10111010000000000000000
Checksum is: 1000101101011000
FinalCodeword is: 10111011000101101011000
Test Error detection 0(yes) 1(no) ? : 0
Enter position where error is to inserted: 3
Errorneous data: 10101011000101101011000
Error detected.


Enter poly: 1011101
Generating Polynomial is: 10001000000100001
Modified t[u] is:  10111010000000000000000
Checksum is: 1000101101011000
FinalCodeword is: 10111011000101101011000
Test Error detection 0(yes) 1(no) ? : 1
No Error Detected.
```

Program – 7

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

C Code: (P7.c)

```
#include <stdio.h>
#include <stdlib.h>
void rout_table();
int d[10][10], via[10][10];
int i, j, k, l, m, n, g[10][10], temp[10][10], ch, cost;
int main()
{
   system("clear");
   printf("Enter the value of number of nodes: ");
   scanf("%d", &n);
   rout_table();
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         temp[i][j] = g[i][j];
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         via[i][j] = i;
   while (1)
   {
      for (i = 0; i < n; i++)
      {
         for (j = 0; j < n; j++)
         {
            if (d[i][j])
               for (k = 0; k < n; k++)
               {
                  if (g[i][j] + g[j][k] < g[i][k])
                  {
                     g[i][k] = g[i][j] + g[j][k];
                     via[i][k] = j;
                  }
               }
         }
      }
```

```
        for (i = 0; i < n; i++)
        {
            printf("Table for router %c:\n", i + 97);
            for (j = 0; j < n; j++)
                printf("%c :: %d via %c\n",j + 97, g[i][j], via[i][j] + 97);
        }
        break;
    }
}
void rout_table()
{
    printf("\nEnter the routing table:\n\t");
    for (i = 1; i <= n; i++)
        printf("%c\t", i + 96);
    printf("\n");
    for (i = 0; i < n; i++)
    {
        printf("%c ", i + 97);
        for (j = 0; j < n; j++)
        {
            scanf("%d", &g[i][j]);
            if (g[i][j] != 999)
                d[i][j] = 1;
        }
    }
}
```

Output:

```
Enter the value of number of nodes: 4
Enter the routing table:
        a       b       c       d
a 0 6 7 999
b 6 0 999 3
c 7 999 0 5
d 999 3 5 0
Table for router a:
a :: 0 via a
b :: 6 via a
c :: 7 via a
d :: 9 via b
```

Table for router b:
a :: 6 via b
b :: 0 via b
c :: 8 via d
d :: 3 via b
Table for router c:
a :: 7 via c
b :: 8 via d
c :: 0 via c
d :: 5 via c
Table for router d:
a :: 9 via b
b :: 3 via d
c :: 5 via d
d :: 0 via d

Program – 8

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Link state algorithm.

C Code: (P8.c)

```
#include<stdio.h>
void dij(int n,int cost[10][10],int source,int dest,int d[],int p[])
{
   int i,j,u,v,min,s[10];
   for(i=0;i<n;i++)
   {
      d[i]=cost[source][i];
      s[i]=0;
      p[i]=source;
   }
   s[source]=1;
   for(i=1;i<n;i++)
   {
      min=999;
      u=-1;
      for(j=0;j<n;j++)
      {
         if(s[j]==0)
         {
            if(d[j]<min)
            {
               min=d[j];
               u=j;
            }
         }
      }
      if(u==-1)
         return;
      s[u]=1;
      if(u==dest)
         return;
      for(v=0;v<n;v++)
      {
         if(s[v]==0)
```

```
            {
               if(d[u]+cost[u][v]<d[v])
               {
                  d[v]=d[u]+cost[u][v];
                  p[v]=u;
               }
            }
         }
      }
}
void print_path(int source,int destination,int d[],int p[])
{
   int i;
   i=destination;
   while(i!=source)
   {
      printf("%d<-",i);
      i=p[i];
   }
   printf("%d=%d\n",i,d[destination]);
}
int main()
{
   int cost[10][10],n,d[10],p[10],i,j;
   printf("Enter the number of nodes in the network\n");
   scanf("%d",&n);
   printf("Enter the cost n between every nodes\n");
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         scanf("%d",&cost[i][j]);
   printf("enter the source node\n");
   scanf("%d",&i);
   for(j=0;j<n;j++)
   {
      dij(n,cost,i,j,d,p);
      if(d[j]==999)
         printf("%d is not reachable from %d\n",j,i);
      else if(i!=j)
         print_path(i,j,d,p);
   }
```

```
    return 0;
}
```

Output:

```
Enter the number of nodes in the network
4
Enter the cost n between every nodes
0 6 7 999
6 0 999 3
7 999 0 5
999 3 5 0
enter the source node
1
0<-1=6
2<-3<-1=8
3<-1=3
```

Program – 9

Write a program for congestion control using leaky bucket algorithm.

C Code: (P9.c)

```
#include<stdio.h>
#include<strings.h>
int min(int x,int y)
{
   if(x<y)
      return x;
   else
      return y;
}
int main()
{
   int drop=0,mini,nsec,cap,count=0,i,inp[25],process;
   system("clear");
   printf("Enter The Bucket Size\n");
   scanf("%d",&cap);
   printf("Enter The Operation Rate\n");
   scanf("%d",&process);
   printf("Enter The No. Of Seconds You Want To Stimulate\n");
   scanf("%d",&nsec);
   for(i=0;i<nsec;i++)
   {
      printf("Enter The Size Of The Packet Entering At %d sec\n",i+1);
      scanf("%d",&inp[i]);
   }
   printf("\nSecond|Packet Recieved|Packet Sent|Packet Left|Packet
Dropped|\n");
   printf("-------------------------------------------------------------------------\n");
   for(i=0;i<nsec;i++)
   {
      count+=inp[i];
      if(count>cap)
      {
         drop=count-cap;
         count=cap;
      }
```

```
        printf("%d",i+1);
        printf("\t%d",inp[i]);
        mini=min(count,process);
        printf("\t\t%d",mini);
        count=count-mini;
        printf("\t\t%d",count);
        printf("\t\t%d\n",drop);
        drop=0;
    }
    for(;count!=0;i++)
    {
        if(count>cap)
        {
            drop=count-cap;
            count=cap;
        }
        printf("%d",i+1);
        printf("\t0");
        mini=min(count,process);
        printf("\t\t%d",mini);
        count=count-mini;
        printf("\t\t%d",count);
        printf("\t\t%d\n",drop);
    }
}
```

Output:

```
Enter The Bucket Size
5
Enter The Operation Rate
2
Enter The No. Of Seconds You Want To Stimulate
3
Enter The Size Of The Packet Entering At 1 sec
5
Enter The Size Of The Packet Entering At 2 sec
4
Enter The Size Of The Packet Entering At 3 sec
3
```

Second|Packet Recieved|Packet Sent|Packet Left|Packet Dropped|

---

1 5 2 3 0

2 4 2 3 2

3 3 2 3 1

4 0 2 1 0

5 0 1 0 0