6. a. Bit Stuffing

Program:

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

int main()
{
   int a[20],b[30],i,j,k,count,n;
   printf("Enter frame size: ");
   scanf("%d",&n);
   printf("Enter the frame in the form of 0 and 1 : ");
   for(i=0; i<n; i++)
      scanf("%d",&a[i]);
   i=0;
   count=1;
   j=0;
   while(i<n)
   {
      if(a[i]==1)
      {
         b[j]=a[i];
         for(k=i+1; a[k]==1 && k<n && count<5; k++)
         {
            j++;
            b[j]=a[k];
            count++;
            if(count==5)
            {
               j++;
               b[j]=0;
            }
            i=k;
         }
      }
      else
      {
         b[j]=a[i];
      }
      i++;
      j++;
   }
   printf("After Bit Stuffing : ");
   for(i=0; i<j; i++)
      printf("%d",b[i]);
   printf("\n");
   return 0;
}
```

Output:

```
Enter frame size: 10
Enter the frame in the form of 0 and 1 : 1 0 1 0 1 1 1 1 1 1
After Bit Stuffing : 10101111101
```

b. Character Stuffing

Program:

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

int main()
{
  char a[30], fs[50] = " ", t[3], sd, ed, x[3], s[3], d[3], y[3];
  int i, j, p = 0, q = 0;
  printf("Enter characters to be stuffed: ");
  scanf("%s", a);
  printf("Enter a character that represents starting delimiter: ");
  scanf(" %c", &sd);
  printf("Enter a character that represents ending delimiter: ");
  scanf(" %c", &ed);
  x[0] = s[0] = s[1] = sd;
  x[1] = s[2] = '\0';
  y[0] = d[0] = d[1] = ed;
  d[2] = y[1] = '\0';
  strcat(fs, x);
  for(i = 0; i < strlen(a); i++)
  {
    t[0] = a[i];
    t[1] = '\0';
    if(t[0] == sd)
      strcat(fs, s);
    else if(t[0] == ed)
      strcat(fs, d);
    else
      strcat(fs, t);
  }
  strcat(fs, y);
  printf("After stuffing: %s", fs);
  printf("\n");
  return 0;
}
```

Output:

```
Enter characters to be stuffed: goodday
Enter a character that represents starting delimiter: t
Enter a character that represents ending delimiter: c
After stuffing:  tgooddayc
```

7. Distance Vector Routing Algorithm

Program:

```
#include<stdio.h>

int dist[50][50],temp[50][50],n,i,j,k,x;

void dvr()
{
 for (i = 0; i < n; i++)
   for (j = 0; j < n; j++)
     for (k = 0; k < n; k++)
       if (dist[i][k] + dist[k][j] < dist[i][j])
       {
         dist[i][j] = dist[i][k] + dist[k][j];
         temp[i][j] = k;
       }

       for(i=0;i<n;i++)
       {
         printf("\n\nState value for Router %d is:",i+1);
         for(j=0;j<n;j++)
           printf("\t\nNode %d via %d Distance %d",j+1,temp[i][j]+1,dist[i][j]);
       }
       printf("\n\n");
}

int main()
{
 printf("Enter the number of nodes : ");
 scanf("%d",&n);
 printf("Enter the distance matrix :\n");
 for(i=0;i<n;i++)
 {
   for(j=0;j<n;j++)
   {
     scanf("%d",&dist[i][j]);
     dist[i][i]=0;
     temp[i][j]=j;
   }
 }
 dvr();
 return 0;
}
```

Output:

```
Enter the number of nodes : 3
Enter the distance matrix :
0 2 7
2 0 1
7 1 0

State value for Router 1 is:
Node 1 via 1 Distance 0
Node 2 via 2 Distance 2
Node 3 via 2 Distance 3
```

```
State value for Router 2 is:
Node 1 via 1 Distance 2
Node 2 via 2 Distance 0
Node 3 via 3 Distance 1

State value for Router 3 is:
Node 1 via 2 Distance 3
Node 2 via 2 Distance 1
Node 3 via 3 Distance 0
```

8. Stop & Wait Flow Control Protocol

Program:

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

int main ()
{
  int i, j, noframes, x, x1 = 10, x2;
  printf("Enter the no. of frames: ");
  scanf("%d",&noframes);
  i = 1;
  j = 1;
  printf ("\nNumber of frames is %d", noframes);
  while (noframes > 0)
   {
     printf ("\nSending frame %d", i);
     srand (x1++);
     x = rand () % 10;
     if (x % 2 == 0)
     {
            for (x2 = 1; x2 < 2; x2++)
            {
                     printf ("\nWaiting for %d seconds", x2);
                     sleep (x2);
            }
            printf ("\nSending frame %d", i);
            srand (x1++);
            x = rand () % 10;
     }
     printf ("\nAcknowledgement for frame %d", j);
     noframes -= 1;
     i++;
     j++;
   }
  printf ("\nEnd of stop and wait protocol");
  printf (“\n”);
  return 0;
}
```

Output:

```
Enter the no. of frames: 10

Number of frames is 10
Sending frame 1
```

```
Waiting for 1 seconds
Sending frame 1
Acknowledgement for frame 1
Sending frame 2
Waiting for 1 seconds
Sending frame 2
Acknowledgement for frame 2
Sending frame 3
Waiting for 1 seconds
Sending frame 3
Acknowledgement for frame 3
Sending frame 4
Waiting for 1 seconds
Sending frame 4
Acknowledgement for frame 4
Sending frame 5
Waiting for 1 seconds
Sending frame 5
Acknowledgement for frame 5
Sending frame 6
Waiting for 1 seconds
Sending frame 6
Acknowledgement for frame 6
Sending frame 7
Waiting for 1 seconds
Sending frame 7
Acknowledgement for frame 7
Sending frame 8
Waiting for 1 seconds
Sending frame 8
Acknowledgement for frame 8
Sending frame 9
Waiting for 1 seconds
Sending frame 9
Acknowledgement for frame 9
Sending frame 10
Waiting for 1 seconds
Sending frame 10
Acknowledgement for frame 10
End of stop and wait protocol
```

9. ERROR detecting code using CRC-CCITT (16bit)

Program:

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

int main ()
{
  int i, j, keylen, msglen;
  char data[100], key[30], temp[30], quot[100], rem[30], key1[30];
  printf ("Enter Data: ");
  scanf("%s",data);
  printf ("Enter Key: ");
  scanf("%s",key);
  keylen = strlen (key);
  msglen = strlen (data);
  strcpy (key1, key);
  for (i = 0; i < keylen - 1; i++)
          data[msglen + i] = '0';
  for (i = 0; i < keylen; i++)
    temp[i] = data[i];
  for (i = 0; i < msglen; i++)
  {
          quot[i] = temp[0];
          if (quot[i] == '0')
                  for (j = 0; j < keylen; j++)
                          key[j] = '0';
                  else
                          for (j = 0; j < keylen; j++)
                                  key[j] = key1[j];
                          for (j = keylen - 1; j > 0; j--)
                          {
                                  if (temp[j] == key[j])
                                          rem[j - 1] = '0';
                                  else
                                          rem[j - 1] = '1';
                          }
                          rem[keylen - 1] = data[i + keylen];
                          strcpy (temp, rem);
  }
  strcpy (rem, temp);
  printf ("\nQuotient is ");
  for (i = 0; i < msglen; i++)
    printf ("%c", quot[i]);
  printf ("\nRemainder is ");
  for (i = 0; i < keylen - 1; i++)
    printf ("%c", rem[i]);
  printf ("\nFinal data is: ");
  for (i = 0; i < msglen; i++)
    printf ("%c", data[i]);
  for (i = 0; i < keylen - 1; i++)
    printf ("%c", rem[i]);
  printf("\n");
  return 0;
}
```

Output:

```
Enter Data: 10101111
Enter Key: 1011

Quotient is 10011010
Remainder is 110
Final data is: 10101111110
```

10. 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\n",bfilled);
        i++;
    }
    return 0;
}
```

Output:

```
Enter the number of packets: 3
Enter packets in the order of their 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
```