1. Write a program for error detection using CRC-CCITT ($x^{16}+x^{12}+x^5+1$).

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

char t[28],cs[28],g[]="10001000000100001";
inta,e,c;

voidxor(){
for(c = 1;c < N; c++)
cs[c] = (( cs[c] == g[c])?'0':'1');
}

voidcrc(){
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 data : ");
scanf("%s",t);
printf("\n----------------------------------------");
printf("\nGeneratng polynomial : %s",g);
   a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("\n----------------------------------------");
printf("\nModified data is : %s",t);
printf("\n----------------------------------------");
crc();
printf("\n CRC checksum is : %s",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];
printf("\n----------------------------------------");
printf("\nFinalcodeword transmitted is : %s",t);
printf("\n----------------------------------------");
printf("\nTest error detection 0(yes) 1(no)? : ");
scanf("%d",&e);
if(e==0)
   {
do{
```

```
printf("\nEnter the position where error is to be inserted : ");
scanf("%d",&e);
}while(e==0 || e>a+N-1);
t[e-1]=(t[e-1]=='0')?'1':'0';
printf("\n---------------------------------------");
printf("\nErroneous data : %s\n",t);
   }
crc();
for(e=0;(e<N-1) && (cs[e]!='1');e++);
if(e<N-1)
        {
printf("\n CRC checksum is : %s",cs);
printf("\nError detected\n\n");
        }
else
        {
printf("\n CRC checksum is : %s",cs);
printf("\nNo error detected\n\n");
        }
printf("\n---------------------------------------\n");
return 0;
}
```

Output 1:

```
Enter data : 101
---------------------------------------
Generatngpolynomial : 10001000000100001
---------------------------------------
Modified data is : 1010000000000000000
---------------------------------------
 CRC checksum is : 0101000010100101
---------------------------------------
Final codeword transmitted is : 1010101000010100101
---------------------------------------
Test error detection 0(yes) 1(no)? : 0
Enter the position where error is to be inserted : 3
---------------------------------------
Erroneous data : 1000101000010100101
CRC checksum is : 0001000000100001
Error detected
---------------------------------------
```

Output 2:

```
Enter data : 101
---------------------------------------
Generatngpolynomial : 10001000000100001
---------------------------------------
Modified data is : 1010000000000000000
```

```
---------------------------------------
 CRC checksum is : 0101000010100101
---------------------------------------
Final codeword transmitted is : 1010101000010100101
---------------------------------------
Test error detection 0(yes) 1(no)? : 1

 CRC checksum is : 0000000000000000
No error detected
---------------------------------------
```

2. Write a Program in C/ C++ for hamming code generation for error detection/correction

```
#include<stdio.h>

void main() {
int data[10];
intdataatrec[10],c,c1,c2,c3,i;

printf("Enter 4 bits of data one by one\n");
scanf("%d",&data[0]);
scanf("%d",&data[1]);
scanf("%d",&data[2]);
scanf("%d",&data[4]);

   //Calculation of even parity
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("\nEncoded 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",data[i]);

printf("\nData received : ");
for(i=0;i<7;i++)
printf("%d",dataatrec[i]);

printf("\nCorrect message is\n");

      //if errorneous bit is 0 we complement it else vice versa
if(dataatrec[7-c]==0)
dataatrec[7-c]=1;
else
dataatrec[7-c]=0;

for (i=0;i<7;i++) {
printf("%d",dataatrec[i]);
      }
   }
}
```

Output 1:

```
Enter 4 bits of data one by one
1
0
1
0

Encoded data is
1010010

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

No error while transmission of data
```

Output 2:

```
Enter 4 bits of data one by one
1
0
1
```

0

Encoded data is
1010010

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

Error on position 3
Data sent : 1010010
Data received : 1010110
Correct message is1010010

Leaky Bucket

```
#include<stdio.h>

int main(){
    int incoming, outgoing, buck_capacity, n, store = 0;
    printf("Enter bucket capacity, outgoing rate and no of inputs: ");
    scanf("%d %d %d", &buck_capacity, &outgoing, &n);

    while (n != 0) { //loop over total number of inputs
        printf("Enter the number of incoming packets: ");
        scanf("%d", &incoming);
        printf("Incoming packet size %d\n", incoming);
  if((incoming-outgoing) <= (buck_capacity-store)) //it is possible to send without dropping
{
int sent = outgoing>=incoming?incoming:outgoing; //if incoming is more than outgoing, total
sent will be outgoing rest buff
if(sent < outgoing && store != 0)  //if incoming<outgoing, we can add values from the store to
be sent till op cap is reached
{
int remaining = outgoing-sent;
while(remaining > 0 && store != 0) //keeps adding one to sent until we run out of op cap or
nothing left in buff
{
remaining -= 1;
store -= 1;
sent += 1;
}
```

```
}
printf("%d packets sent out\n", sent);
if(outgoing<incoming)
{
store = store + (incoming - outgoing);
}
}
else //packets need to be dropped
{
int dropped = (incoming-outgoing)-(buck_capacity-store); //excess packets are the ones that
after sending cant be accomodated in buff
printf("%d packets dropped\n", dropped);
store = buck_capacity;
}
printf("%d out of %d space used in the buffer\n", store, buck_capacity);
        n--;
    }
}
```