

WEEK 13

Program 1

Write a program for error detecting code using CRC-CCITT (16-bits).

Observation:

Write a program for error detecting code using CRC-CCITT (16-bits).

```
#include <stdio.h>
#include <string.h>
#define N strlen(poly)
char data[30];
char check_value[30];
char poly[16];
int data_length, i, j;

void XOR()
{
    for (j=1; j<N; j++)
        check_value[j] = (check_value[j] ^ poly[j] ? '0' : '1');
}

void receiver()
{
    printf("Enter the received data: ");
    scanf("%s", data);
    printf("Data Received: %s", data);
    XOR();
    for (i=0; (i<N-1) && (check_value[i] != '1'); i++);
    if (i<N-1)
        printf("An Error detected\n");
    else
        printf("No error detected\n");
}

void crc()
{
    for (i=0; i<N; i++)
        check_value[i] = data[i];
    do {
```

```

    if (check_value[i] == '1')
        XOR();
    for (j=0; j<N-1; j++)
        check_value[j] = check_value[j+1];
    check_value[j] = data[j+1];
} while (i < data_length + N-1);
}

int main()
{
    printf("Enter data to be transmitted: ");
    scanf("%s", data);
    printf("Enter the divisor polynomial: ");
    scanf("%s", poly);
    data_length = strlen(data);
    for (i = data_length; i < data_length + N-1; i++)
        data[i] = '0';

    printf("Data padded with n-1 zeros: %s", data);
    printf("In CRC value is %s", check_value);
    for (i = data_length; i < data_length + N-1; i++)
        data[i] = check_value[i - data_length];
    printf("Final dataword to be sent: %s", data);
    receiver();
    return 0;
}

```

Output:

Enter the data to be transmitted: 101010
 Enter the divisor polynomial: 1011

Data padded with $N-1$ zeros : 101010000

CRC value is : 001

Final codeword to be sent : 101010001

Enter the received data : 10001000

Error detected.

2) Enter the data to be transmitted : 101100

Enter the divisor polynomial : 1001

Data padded with $N-1$ zeros : 101100000

CRC value is : 001

Final codeword to be sent : 101100001

Enter the received data : 101100001

No error detected

CODE:

```
#include<stdio.h>
#include<string.h>
#define N strlen(gen_poly)
char data[28];
char check_value[28];
char gen_poly[10];
int data_length,i,j;
void XOR(){
    for(j = 1;j < N;j++)
        check_value[j] = ((check_value[j] == gen_poly[j])?'0':'1');
}
```

```

void receiver(){
    printf("Enter the received data: ");
    scanf("%s", data);
    printf("Data received: %s", data);
    crc();
    for(i=0;(i<N-1) && (check_value[i]!='1');i++);
    if(i<N-1)
        printf("\nError detected\n\n");
    else
        printf("\nNo error detected\n\n");
}

void crc(){
    for(i=0;i<N;i++)
        check_value[i]=data[i];
    do{
        if(check_value[0]=='1')
            XOR();
        for(j=0;j<N-1;j++)
            check_value[j]=check_value[j+1];
        check_value[j]=data[i++];
    }while(i<=data_length+N-1);
}

int main()
{
    printf("\nEnter data to be transmitted: ");
    scanf("%s",data);
    printf("\n Enter the Generating polynomial: ");
    scanf("%s",gen_poly);
    data_length=strlen(data);
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n Data padded with n-1 zeros : %s",data);
    crc();
}

```

```

printf("\nCRC or Check value is : %s",check_value);
for(i=data_length;i<data_length+N-1;i++)
    data[i]=check_value[i-data_length];
printf("\n Final data to be sent : %s",data);
receiver();
return 0;
}

```

OUTPUT:

```

Enter data to be transmitted: 10001000000100001

Enter the Generating polynomial: 1011

Data padded with n-1 zeros : 10001000000100001000
CRC or Check value is : 100
Final data to be sent : 10001000000100001100
Enter the received data: 10001000000100001100
Data received: 10001000000100001100
No error detected

```

```

Enter data to be transmitted: 10001000000100001

Enter the Generating polynomial: 1011

Data padded with n-1 zeros : 10001000000100001000
CRC or Check value is : 100
Final data to be sent : 10001000000100001100
Enter the received data: 10010000000100001100
Data received: 10010000000100001100
Error detected

```


Program 2

Write a program for congestion control using Leaky bucket algorithm.

Observation:

Write a program for congestion control using Leaky Bucket algorithm

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

```
void main()
```

```
{
```

```
int b_size, d_rate, in_d_rate, rem_b_size;
```

```
printf("Enter the bucket size: \n");
```

```
scanf("%d", &b_size);
```

```
rem_b_size = b_size;
```

```
printf("Enter the outgoing data rate: \n");
```

```
scanf("%d", &d_rate);
```

```
while(1)
```

```
{
```

```
printf("Enter the size of incoming packet: \n");
```

```
scanf("%d", &in_d_rate);
```

```
if (in_d_rate <= b_size)
```

```
{
```

```
rem_b_size = rem_b_size - in_d_rate;
```

```
rem_b_size += d_rate;
```

```
printf("Data packet is accepted \n");
```

```
printf("Remaining space in bucket is - %d \n", rem_b_size);
```

```
printf("\n");
```

```
}
```

```
else
```

```
printf("Data packet is dropped because the bucket packet size is more than the remaining bucket space");
```

```
printf("\n");
```

```
}
```

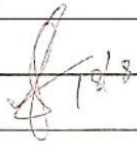
```
}
```

Output:

Enter the bucket size: 5000
Enter the outgoing data rate: 200
Enter the incoming packet: 3000
Data packet is accepted.

Remaining space in bucket is 2000

Enter the size of incoming packet: 3000
Data packet is dropped because the packet size is more than the remaining bucket space.

 12/8

CODE:

```
#include<stdio.h>
void main()
{
    int b_size,d_rate,in_d_rate,rem_b_size;
    printf("Enter the bucket size:\n");
    scanf("%d",&b_size);
    rem_b_size=b_size;
    printf("Enter the outgoing data rate:\n");
    scanf("%d",&d_rate);
    while(1)
    {
```

```

printf("Enter the size of incoming packet\n");
scanf("%d",&in_d_rate);
if(in_d_rate<=b_size)
{
    if(in_d_rate<=rem_b_size)
    {
        rem_b_size=rem_b_size-in_d_rate;
        rem_b_size=rem_b_size+d_rate;
        printf("Data packet is accepted\n");
        printf("Remaining space in bucket is.... %d\n",rem_b_size);
        printf("\n");
    }
    else{
        printf("Data packet is dropped because the bucket size is less than the packet
size\n");
        printf("\n");
    }
}
}
}
}
}

```

OUTPUT:

```

Enter the bucket size:
5000
Enter the outgoing data rate:
200
Enter the size of incoming packet
3000
Data packet is accepted
Remaining space in bucket is.... 2200

Enter the size of incoming packet
2500
Data packet is dropped because the bucket size is less than the packet size

Enter the size of incoming packet

```