Output

```
Enter the data bits: 1001001000100100
Enter the key (divisor): 1101
Encoded Data: 100100100010010111

Decoding the encoded data...
Remainder after decoding: 000
No error detected in received data

=== Code Execution Successful ===
```

EXPERIMENT - 13

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

```
def xor(a,b):
    result = []
    for i in range(1, len(b)):
        if a[i] == b[i]:
            result.append('0')
        else:
            result.append('1')
    return ''.join(result)

def mod2div(dividend, divisor):
    pick = len(divisor)
    tmp = dividend[0:pick]
    while pick < len(dividend):
        if tmp[0] == '1':
            tmp = xor(divisor, tmp) +
                  dividend[pick]
        else:
            tmp = xor('0'*pick, tmp) +
                  dividend[pick]
        pick += 1
    if tmp[0] == '1':
        tmp = xor(divisor, tmp)
    else:
        tmp = xor('0'*pick, tmp)
    checkword = tmp
    return checkword

def encode(data, key):
    l_key = len(key)
    appended_data = data + '0'*(l_key-1)
    remainder = mod2div(appended_data, key)
```

```
    codeword = data + remainder
    print("Remainder : ", remainder)
    print("Encoded Data : Data + remainder", codeword)

    return codeword
def decode_data(encoded_data, key):
    remainder = mod2div(encoded_data, key)
    print("Remainder after decoding : ", remainder
    if '1' not in remainder :
        print("No error detected in received
              data")

    else:
        print("Error detected in received
              data")
data = "100100100010010"
key = "1001"
encoded_data = encode(data, key)
decoded_data = decode_data(encoded_data, key)
```

OUTPUT :

Remainder = 11

encoded_data (data + remainder) =
100100100010010011

Remainder after decoding = 000

No error detected in received data.