EXPERIMENT 5

5. Write a Program to implement on a data set of characters the three CRC polynomials CRC 12, CRC 16 and CRC CCITT.

PROGRAM:

```
def xor(x, y):
 ans = ""
  for i in range(1, len(y)):
if x[i] == y[i]:
     ans += '0'
     else:
 ans += '1'
return ans

def divide(dividend, divisor):
 a = len(divisor)
  temp = dividend[0:a]
  while a < len(dividend):
   if temp[0] == '1':
    temp = xor(divisor, temp) + dividend[a]
     else:
 temp = xor('0' * a, temp) + dividend[a]
 a += 1
 if temp[0] == '1':
  temp = xor(divisor, temp)
     else:
 temp = xor('0' * a, temp)
     return temp

keys      =      ['1100000001111',      '11000000000000101',
'10001000000100001']

print("Choose the CRC")
print("1. CRC - 12")
print("2. CRC - 16")
print("3. CRC - CCITT ")
n = int(input())
send = input("Enter the string of code word of binary data
bits of 0's and 1's to be sent from the sender: ")
rec = input(" Enter the string of code word of binary data
received at the receiver side: ")
key = keys[n - 1]
```

```
# encoding sender side
length = len(key)
send1 = send + '0' * (length - 1)
rem = divide(send1, key)

# decoding receiver side
ans = divide(rec, key)
if (ans == '0' * (len(key) - 1)):
 print("no error")
else:
 print("frame error")
```

OUTPUT:

```
Choose the CRC


1. CRC - 12

2. CRC- 16

3. CRC- CCITT

2

Enter the string of code word of binary data bits of 0's and
1's to be sent from the sender: 101110111010101

Enter the string of code word of binary data received at the
receiver side: 10111011101010100110011111011

no error


Choose the CRC


1. CRC- 12

2. CRC- 16

3. CRC- CCITT

1
```

```
Enter the string of code word of binary data bits of 0's and
1's to be sent from the sender: 1010101

Enter the string of code word of binary data received at the
receiver side: 10101010010000000010


no error
```

EXPERIMENT 11

11. Write a Program to implement Distance vector routing algorithm by obtaining routing table at each node (Take an example subnet graph with weights indicating delay between nodes).

PROGRAM:

```
# IMPLEMENTATION OF DISTANCE VECTOR:
INFINITY = 10000
length = [[0 for _ in range(10)] for _ in range(10)]
path = [[0 for _ in range(10)] for _ in range(10)]
se = [0] * 10
adj = [] s = c = 0

n = int(input("Enter No of Routers: "))
print("Enter Adjacency Matrix")
for i in range(n):
 adj.append(list(map(int, input().split())))

# Initialization Part for i in range(n):
for j in range(n):
     if adj[i][j] == 0 and i != j:
     length[i][j] = INFINITY
     path[i][j] = 0
 else:
  length[i][j] = adj[i][j]
  path[i][j] = j
```

```
 if i == j:
  path[i][j] = 30

# Iteration Part
c = 1
while c:
 c = 0
 for s in range(n):
  for j in range(n):
   if adj[s][j]:
    for i in range(n):
 if (length[s][j] + length[j][i]) <
length[s][i]:

 length[s][i]= length[s][j] +length[j][i]
  path[s][i] = j
for s in range(n):
 for i in range(n):
  if length[s][i] == INFINITY:
  c += 1
print("\nRouting table\n\n")
for i in range(65, 65 + n):
 print("  ", chr(i), "   ", end=' ')
 print("\n-------------------------------------------")

for i in range(n):
 print(chr(i + 65), end=' ')
 for s in range(n):
```

```
  print(" %3d%3c |"%(length[s][i], path[s][i] + 65), end='')
 print()
```

OUTPUT:

Enter No of Routers: 4

Enter Adjacency Matrix

0 6 0 1

6 0 2 4

0 2 0 1

1 4 1 0

Routing table

A		B		C		D	
A	0 _	4	D	2	D	1	A
B	5 D	0	_	2	B	3	C
C	2 D	2	C	0	_	1	C
D	1 D	3	C	1	D	0	_