Ex. no: 6
Date: 7/8/24

Aim: To write a program to implement error detection and correction using Hamming code concept. Make a test run to input data stream and verify error correction feature.

Error Correction at Data Link Layer:

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when data is transmitted from the sender to receiver. It is a technique developed by R.W. Hamming for error correction.

Creates sender program

1) Input to sender file should be a text of any length. Program should convert the text to binary.
2) Apply hamming code concept on the binary data and add redundant bits to it.
3) Save this output in a file called channel.

Create a reciever program with below features.

1) Reciever program should read the input from channel file.
2) Apply hamming code on binary data to check for errors.
3) If there is an error, display the position of error.
4) Else remove the redundant bits and convert the binary data to ASCII and display o/p

Code

```
string = input ("Enter string:")
s = ''.join (format (ord(i), '08b') for i in string)
rb = 0
for i in range (len (s)):
    if (2**i >= len(s) + i + 1):
        rb = i
        break
m = len(s) + rb
l = []
pos = []
c = 0
s = s[::-1]
for i in range (rb):
    pos.append (2**i)
for i in range (m):
```

```
        if (i+1) in pos:
            l.insert(i, 'p')
        else:
            l.insert(i, int(s[c]))
            c+=1
print("parity bits positions:", pos)
print("Encoded data with 'p' as parity position:", l)
for p in pos:
    count=0
    i=p-1
    while i<n:
        count+= l[i:i+p].count(1)
        i+=2*p
    l[p-1]=1 if count%2!=0 else 0
print("final encoded data with parity bits:", l[::-1])
b=open("code.txt","w")
b.write(str(l[::-1]))
b.close()

b=open("Original.txt","w")
b.write(str(l[::-1]))
b.close()
```

```
f = open("code.txt", "r")
x = f.readline()
print("B code:", x)
g = open("Original.txt", "r")
y = g.readline()
print("O code:", y)
l = x.strip("[").strip("]").strip('],').split(",")
for i in l:
    i.strip(" ")
l = []
pos = []
rb = 0
for i in range(len(l)):
    if (2**i >= len(l) + i + 1):
        rb = i
        break
for i in range(rb):
    pos.append(2**i)
print("Positions:", pos)
q = []
for i in l[::-1]:
    if (i == "," or i == ","):
        q.append(1)
```

```
        else:
            q.append(0)
for p in pos:
    count=0
    i=p-1
    while i< len(q):
        count+=q[i:i+p].count(1)
        i+=2*p
    c.append(0) if count%2==0 else c.append(1)
print("Binary:",c[::-1])
change=0
q=q[::-1]
for i in range(len(c[::-1])):
    change+=c[i]*2**i
q[change-1]=0 if q[change-1]==1 else 1
print("Error:",change)
print("Corrected code",q)
```

Output

```
Enter string: b
parity bit positions: [1,2,4,8]
Final encoded data : [0,1,1,0,0,1,0,0,1,0,1,0]
```

Encode : [0,1,1,0,0,1,0,1,1,0,1,0]

Binary: [0, 0, 1, 0, 1]

Error: 5

Corrected Code: [0,1,1,0,1,1,0,1,1,0,1,0]

Result:

Enter input String: hello

Generated Hamming Code: 110111011000011100101011011000111011000111

~~0001~~ 01111

Hamming Code with error: 111111011000011100101011011000111p11

000111010111

Error detected at position: 3

Corrected Hamming Code: 110111011000011100101011011000111011

000111011111

Corrected bit at position 3:0

Result

Thus the Hamming code detection and correction is applied and output is verified.