Ex: No: 6
12/9/24

Error Correction at Data Link Layer

Aim:

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 Data Link Layer:

Hamming code is a set of error correction code that can be used to detect and correct the errors that can occurs when the data is transmitted from the sender to the receiver.

Create sender program with below features:

* Input to sender file should be a text of any length program should convert the text to binary.
* Apply programming code concept on the binary data and add redundant bits to it.
* Save this output in file called channel.

Create a receiver program with below feature.

For Receiver:

* Receiver program should read the input from the channel file.
* Apply hamming code on the binary data to check for errors.

* Apply hamming code on the binary data to check for errors.

* If there is an error, display the position of the error.

Sender

```
String = input ("Enter String")
S = join(format(ord(i), '08b') for i in st
rb = 0
for i in range (1, len(s)):
    if (2**i) >= len(s) + i + 1):
        rb = i
        break;
m = len(s) + rb
l = []
p = []
c = 0
s = s[::-1]
for i in range (rb)
    p.append(2**i);
for i in range (m):
    if (i+1) in p:
        b.insert(i, 'p')
    else:
        b.insert(i, int(s[c]))
        c += 1
print ("parity bit position:", p)
print ("intial encoded data: with p at
    ri sit:")
```

```
    i=0
    i=p-1
    while i<m:
      count += l[i:i+p], count(1)
      i += 2*p
    l[p-1] = 1 if count%2 != 0 else 0
    print("Find encoded data with parity bit:",l)
    f = open("code.txt","w")
    f = open("original.txt","w")
    f = write(str(l[::-1]))
```

Receive pg:

```
    f=open("code.txt","r")
    t=f.readline()
    print("Encode:",t)
    e = open("original real.txt","r")
    y= e.readline()
    Print("code:",y)
    l = t.strip("[").strip("]").strip("\n
        strip(",")
    for i in l:
      i.strip(" ")
    l = []
    p = []
    nb = 0

for i in range(len(l)):
  if (2**i >= len(l)+i+1):
      nb = p
      break
```

```
for i in range(ab):
    p.append(a**i)
print("position:",p)
q=[]
for i in s[::-1]:
    if(i=='1' or i=='0'):
        q.append(i)
    else:
        q.append(0)
for p in pos
    count=0
i=p-1
while i<len(l):
    count+=q[i:i+p].count('1')
    i+=2*p
(.append(os) if count%2==0 else
(.append('1')
print("BINARY",(s[::-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]
print("Error:",change)
print("correct code:",q)
```