EX: 06 ERROR CORRECTION AT DATA LINK LAYER

DATE: 06-08-24

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 detection & correction feature.

Program:

```
def calcRedundantBits(m):
    # use the formula 2^r >= m+r+1
    # to calculate the no. of redundant
    # bit. Iterate over 0 .... m and
    # return the value that satisfies the
    # equation.

    for i in range(m):
        if (2**i >= m+i+1)
            return i

def posRedundantBits(data, r):
    # Redundancy bits are placed
    at the position which correspond
    to the power of 2
```

```
import math
def calculate_redundant_bits(k):
    for r in range(1,k):
        if(2**r >= r+k+1):
            return r

def positioning_redundant_bit(data,r,k):
    result = []
    j=0
    p=k-1
    result.append('1')
    for i in range(1,k+r+1):
        if(2**j==i):
            j=j+1
            result.append('r')
        else:
            result.append(data[p])
            p=p-1
    return result[::-1], result

def building_ham_code(data,k,r):
    j=0
    c=0
    for t in range(1,k+r+1):
        if(t==2**j):
            i=2**j
            p=t
            c=0
            while(p<k+r+1):
                for h in range(p,p+i):
                    try:
                        if(data[h]=='1'
                        and p<k+r+1):
```

```
            c+=1
        except IndexError:
            break
    p=p+i+1
    if (c%2==0):
        data[i]='0'
    else:
        data[i]='1'
    i=i+1
if (flag==0):
    print('Built hamcode: '+''.join(
                data[::-1]))
return data

def checking_ham_code(data,k,r):
    j=0
    c=0
    for t in range(1,k+r+1):
        if (t==2**j):
            i=2**j
            p=t
            c=0
            while(p<k+r+1):
                for h in range(p,p+i
                    try:
                        if(data[h]=='1'
                        and p<(k+r+1):
                            c+=1
                    except IndexError:
                        break
            p=p+1+i
```

```
        if (cr.2**j==0):
            pos = position_redbit(data,k,r)
            return pos
            break
        j = j+1

def position_redbit(data,k,r):
    j=0
    pos=0
    t = buildingham_code(data,k,r)
    for i in range(1,k+r+1):
        if (i == 2**j):
            pos = (10**j) * int(data[i-1]) + pos
            j = j+1
    print("Error position:" + str(int(str(pos),2)))
    return int(str(pos),2)

c = input("Enter the string that you want to send")
data = ''.join(format(ord(i),'08b') for i in c)
k = len(data)
r = calculate_Redundant_bits(k)
print("The string in Binary form: " + "".join(data))
data, result = positioning_redundant_bit(data,r,len(data))
print("The string after redundant bit is inserted " + ''.join(data))
flag=0
data = building_ham_code(result,k,r)
flag=1
```

```
print("Transmission time....")
n= int(input("Enter the position to change during transmission"))
if(math.sqrt(n) = math.floor(math.sqrt(n))):
    print("you can't change the redundant bit")
else:
    data[n] = '0' if(data[n]=='1') else '1'
    print("After error: "+"".join(data[::-1]))
    h= "".join(data[::])
    pos= checking_ham_code(data,k)
    h= list(h)
    h= p
    h[pos-1] = '0' if(h[pos-1]=='1') else '1'
    print("The corrected Error string: "+ ''.join(h[::-1])
```

Correct output:

Enter the string you want to send: red

The string in binary form:

0 111 0 0100 1100 1010 1001 0010 0

The string after redundant bit:

0 1 1 1 0 1 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0

NEW MAHADEV GOLD
Page
Date

n time ... ")
nter the position
xing transmission")
math floor(math.
):
it change the
nt bit").

if (data[n] == '1')
else '1'
error: " + "a".join
(data[::-1]))
ta[::])
am code (data, k, r)

(h[pos-1] == '1')
else '1'
rected Error
+ ''.join(h[::-1])

u want to

orm:
1100101011001100

ant bit:

1100 0100 000

NEW MAHADEV GOLD
Page
Date

Built hamcode: 0111 001 0011000 1010110
10 100000

transmission time .....

Enter the position you want to change:
3

After Error: 0111 001001 1000 1010110100100 00

Error position: 3

Corrected string: 0111 001 0011000 1010110 1
0100000

Result:

The code of ~~has~~ building & checking the hammicode code is successfully written and the output is verified.

8 16/8/24