Practical-6

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 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 the data is transmitted from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

Create sender program with below features.

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 receiver program with below features

1. Receiver program should read the input from Channel file.
2. Apply hamming code on the binary data to check for errors.
3. If there is an error, display the position of the error.
4. Else remove the redundant bits and convert the binary data to ascii and display the output.

Student observation:-

Write the code here:

```
    def calcRedundantBits(m):

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


def posRedundantBits(data, r):
  j = 0
  k = 1
  m = len(data)
  res = ''
```

```
    for i in range(1, m+r+1):
        if(i == 2**j):
            res = res + '0'
            j += 1
        else:
            res = res + data[-1 * k]
            k += 1
     return res[::-1]

def calcParityBits(arr, r):
    n = len(arr)
      for i in range(r):
        val = 0
        for j in range(1, n + 1):

                    if(j & (2**i) == (2**i)):
               val = val ^ int(arr[-1 * j])



            arr = arr[:n-(2**i)] + str(val) + arr[n-(2**i)+1:]
    return arr

def detectError(arr, nr):
    n = len(arr)
    res = 0

      for i in range(nr):
        val = 0
        for j in range(1, n + 1):
            if(j & (2**i) == (2**i)):
                val = val ^ int(arr[-1 * j])

        res = res + val*(10**i)

    return int(str(res), 2)

data = '1011001'
m = len(data)
r = calcRedundantBits(m)
arr = posRedundantBits(data, r)
arr = calcParityBits(arr, r)
print("Data transferred is " + arr)
arr = '11101001110'
print("Error Data is " + arr)
correction = detectError(arr, r)
if(correction==0):
    print("There is no error in the received message.")
else:
    print("The position of error is ",len(arr)-correction+1,"from the left")
```

Input:-
 data = '1011001'

Output:
```
Data transferred is 10101100101
```

Result:
Thus error detection and error correction using hamming code has been implemented successfully.