Ex. No: 6
Date: 10/8/24

Error Detection & Correction using Hamming Code

Aim: Write a program to implement error detection & correction using hamming code. Make a test run to input data and verify error correction feature.

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 hamming code concept on binary data and add redundant bits to it.

-> Save this output in a file called Channel

-> Create a receiver program with below features

-> receiver program should read the input from Channel file

-> apply hamming code on binary data to check for errors.

If there is an error ... error.
else remove the ... the binary data to ...

Student Observation:

Codes:

```
import math
def calculate_...
    for r in ...
        if (2 ** ...
            return
def positioning...
    result = ...
    j = 0
    p = k - 1
    result.a...
    for i in ...
        if (...
        else
    return
def buildin...
```

→ If there's an error display the position of error.

→ else remove the redundant bits & convert the binary data to ASCII & display output

Student observation:

Code:

```
import math
def calculate_redundant_bits(k):
    for r in range(1, 10):
        if (2**r >= r+k+1):
            return r

def positioning_redundancy_bit(data, r, k):
    result = []
    j = 0
    p = k-1
    result.append('')
    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
```

```
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+c+1):
                        c+=1
                except IndexError:
                    break
            p=p+i+1
        if(c%2==0):
            data[i]='0'
        else
            data[i]='1'
        j=j+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 index error
        break;
    p = p+1+i
if (c%2 != 0):
    pos = position - redbit (data, k, r)
    return pos
    break
j = j+1
def position - redbit (data, k, r):
    j = 0
    pos = 0
    t = building ham-code (data, k, r)
    for i in range (1, k+r+1):
        if (i==2**j):
        pos = (10**j) * int (data[i]) + pos
        j = j+1
```

```
print("error position" + str(int(pos)))
c = input("enter the string that you want
    to send")
data = ''.join(format(ord(i), '08b') for i in
data = list(data)
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(k, r)
flag=1
Print("transmission time...")
n = int(input("Enter the position to change
during transmision"))
if (math.sqrt(n) = math.floor(math.
    sqrt(n)))
print("you can't change the redundant bit")
```

```
else:
    data[n]='0'
else:
print("after
h = ''.join(
pos = checking_
h = print(h)
h[pos-1]=
print("The
    join(h
output:
Enter the
The string
The string
0 1
Built ham
transmision
Enter th
After re
Error
```

```
else:
    data[n]='0' if [data[n]=='1'] else '1'

~~else '1'~~
print("After error "+"".join(data[1:-1])
h="".join(data[::])
pos=checking_ham_code(data,k,r)
h=list(h)
h[pos-1]='0' if (h[pos-1]='1') else '1'

print("The corrected error string :"+"".
    join(h[1:-1]);
```

output:

Enter the string you want to send: red

The string in binary form: 01110010011001010110
00100

The string after redundant bit:

0111010011001101011101010011
0111010011001101011101010011

Built hamcode : 011100100110001010110101000000

transmission time . . . . .

Enter the position you want to change : 3

After error : 011100100110001010110101001000100

Error position : 3

Corrected string : 0111001000000001010,
0100000

Result:

The code of building & checking the hammingcode is succesfully written and the output is verified.

Ex.NO:7
Date: 6.9.24

Aim:
write
Control ot
window
frames

Create a
feature
1) Input
2) Input
3) Consi
4) creat
[frame
5) sen
6) wa
7) Rea
8) Che
9) If
send