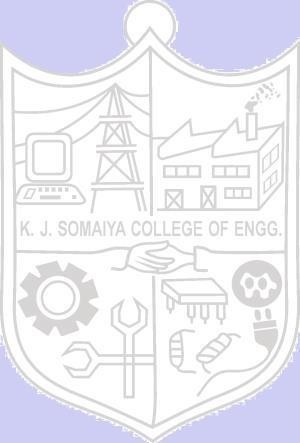
**Experiment No: 05**

 **Title:** Implementation of Error Detection and Correction

(Autonomous College Affiliated to University of Mumbai)

**Batch: B1 Roll No.: 1914078 Experiment No. 05**

**Aim:** To interpret the concept of redundancy in data message for error detection and Correction using Hamming Code

**Resources Used:** Java/ Turbo C **/**Python

**Theory:**

It is often the case that the data retrieved or received is different from the data stored or transmitted, either because the medium is susceptible to damage or because the channel used for transmission is noisy. For example, the data sequence 1101 could be transmitted, but 1001 could be received because the second bit got flipped by the channel. One of the simplest mechanisms to detect whether a single-bit error has occurred is by adding a parity bit to the data sequence. In this practical, we will look at a slightly more complex mechanism to detect and correct single bit errors. The mechanism takes as its input a data sequence of 4 bits and encodes it into a data sequence of 7 bits, which are then transmitted. There is structured redundancy in these 7 bits so that the receiver can detect up to two bit errors and can correct a single bit error.

The (7,4) Hamming code was introduced by Richard Hamming in 1950, who was working then at Bell Telephone Labs. Hamming code uses parity bits concept which is added when you prepare hamming code from the given data stream.

Let ‘k’ be the no of data bits

‘r’ be the no of parity bits and

‘n’ be the number of message bits Then

2r > = k + r +1

No of message bits : n = k + r No of parity bits

: r = n - k

Minimum number of Parity bits :-

Given below is a list of minimum no. of parity bits needed for various ranges of ‘k’

information bit.

|  |  |
| --- | --- |
| No of ‘k’ | No of parity bits(r) |
| 2 to 4 | 3 |
| 5 to 11 | 4 |
| 12 to 26 | 5 |
| 27 to 57 | 6 |
| 58 to 120 | 7 |

Consider a data stream of 7 bits 1011101. It requires four redundant bits. These redundant bits are placed at positions1, 2, 4 and 8 a shown below:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| d | d | d | r | d | d | d | r | d | r | r |

These redundant bits are r1, r2, r4 and r8.

Consider even parity and Calculate

r1 = d3 EXOR d5 EXOR d7 EXOR d9 EXOR d11=0

r2 = d3 EXOR d6 EXOR d7 EXOR d10 EXOR d11=0

r4 = d5 EXOR d6 EXOR d7=0

r8 = d9 EXOR d10 EXOR d11 =0

The message to be transmitted is 10101100100. This is the hamming code which is to be transferred to the destination.

The receiver takes the transmission and recalculates the new values of r. Again the same mechanism is implemented at the receiver end using the same set of data bits plus the relevant parity bits ( r )bit for each set .

If we get the values 0000 then no error, else there is error.

For Correction of the one bit in error, find the decimal equivalent of the binary number obtained using parity bits ( r ) and invert the bit in the corresponding position.

**Program and Output:**  **Python Code:**

trans = input("Enter transmission data: ")

rlis = [0]\*4

if(4<=len(trans)<=11):

    new\_str = trans[-len(trans):-4]+"0"+trans[3:6]+"0"+trans[6]+"00"

    rlis[-1] = int(new\_str[-11]) ^ int(new\_str[-9]) ^ int(new\_str[-7]) ^ int(new\_str[-5]) ^ int(new\_str[-3])

    rlis[-2] = int(new\_str[-3]) ^ int(new\_str[-6]) ^ int(new\_str[-7]) ^ int(new\_str[-10]) ^ int(new\_str[-11])

    rlis[-3] = int(new\_str[-5]) ^ int(new\_str[-6]) ^ int(new\_str[-7])

    rlis[-4] = int(new\_str[-9]) ^ int(new\_str[-10]) ^ int(new\_str[-11])

    print(rlis)

    print("The Hamming code is: ",new\_str)

# ---------------------------------------------

rec = input("Enter received data: ")

rlis2 = [0]\*4

if(8<=len(rec)<=15):

    rlis2[-1] = int(rec[-11]) ^ int(rec[-9]) ^ int(rec[-7]) ^ int(rec[-5]) ^ int(rec[-3]) ^ int(rec[-1])

    rlis2[-2] = int(rec[-3]) ^ int(rec[-6]) ^ int(rec[-7]) ^ int(rec[-10]) ^ int(rec[-11])^int(rec[-2])

    rlis2[-3] = int(rec[-5]) ^ int(rec[-6]) ^ int(rec[-7]) ^ int(rec[-4])

    rlis2[-4] = int(rec[-9]) ^ int(rec[-10]) ^ int(rec[-11]) ^ int(rec[-8])

    err = int("".join( [str(x) for x in rlis2]),2)

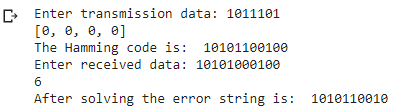
    print(err)

    correctBit = str((int(rec[-err])+1)%2)

    resolved = rec[:-err] + correctBit + rec[-err+1:-1]

    print("After solving the error string is: ",resolved)

**Output:**



**Questions:**

1. **What are the different methods used for error detection?**

**Ans: The most popular Error Detecting Methods are:**

* + - Single parity check
    - Two-dimensional Parity check
    - Checksum
    - Cyclic redundancy check

1. **If the data unit is 111111 and the divisor is 1010. What is the dividend at the Transmitter?**

**Ans:** If divisor is n bit long then we add (n-1) number of 0's in the data unit before division so here divisor is 1010 hence three 0's are added in data unit so the dividend at the transmitter will be - 111111000.

1. **Which layer of the OSI model usually does the function of error detection?**  **Ans:** The **Data Link layer** performs error detection on incoming packets. Networks often use cyclic redundancy check algorithms to find corrupted data at this level. The **Transport layer** handles error recovery.

1. **What arithmetic is used to add data items in checksum calculation?**

**Ans**: One’s complement arithmetic is used to add data items in checksum calculation. In this arithmetic, when a number needs more than n bits, the extra bits are wrapped and added to the number. In this arithmetic, the complement of a number is made by inverting all bits

1. **What is Hamming distance ? What is minimum Hamming distance?**

**Ans:**

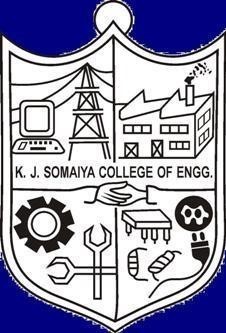
* + The Hamming distance between two equal-length strings of symbols is the number of positions at which the corresponding symbols are different.
  + In a set of strings of equal lengths, the minimum Hamming distance is the smallest Hamming distance between all possible pairs of strings in that set.

**Outcomes:**

Execute their knowledge of computer communication principles, including Error detection and correction, multiplexing, flow control, and error control.

**Conclusion:**

Through this experiment we have learned to interpret the concept of redundancy in data message for Error detection and Correctionusing Hamming Code by writing a computer program for the same and we give the correct code too.



**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in**

**-**

**charge with date**

**References:**

**Books/ Journals/ Websites:**

* + B. A. Forouzan and Firouz Mosharraf, ”Computer Networks ”, A Top-Down Approach, Special Indian Edition 2012, Tata McGraw Hill.
  + Behrouz A Forouzan, Data Communication and Networking, Tata Mc Graw Hill, India,

4th Edition