SVKM’S NMIMS University

Mukesh Patel School of Technology Management and Engineering

**PART A**

**(PART A: TO BE REFFERED BY STUDENTS)**

**Experiment No.07**

**Write a program to generate hamming code**

**A.1—Aim:**

The purpose of this session is to write a programming which can generate the hamming code

**A.2--- Prerequisite:**

Understanding the basics Error Detection and correction techniques

**A.3--- Outcome:**

After successful completion of this experiment students will be able to understand how hamming codes are generated and how error can be detected in hamming code

**A.4--- Procedure**:

Enter 4 data bits

The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
2. All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.   
   Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)  
   Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)  
   Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)  
   Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)  
   Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)  
   Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)  
   etc.
4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

**A.5--- Task:**

1. Write the program to generate hamming code for 4 bit data.

2. Check the output and complete PART B of lab manual

3. Save and close the file and name it as EXP5\_ your Roll no.

**(PART - B)**

|  |  |
| --- | --- |
| Roll. No.: N049 | Name: Tarun Tanmay |
| Sem/Year: V/ third year | Batch: B3 |
| Date of Experiment: 26/08/2020 | Date of Submission: |
| Grade -- |  |

**B.1: Code of performed experiment**

Program:

Text

Description automatically generated

Output:

Text

Description automatically generated

**B.2: Observations and Learning’s:**

The data link layer has the responsibility of error control, which is error detection and correction. Error detecting has many methods like CRC, Simple parity, 2D parity, etc. For error correction, we use Hamming code. In Hamming code, we generate parities based on the data bits and put them in positions that are powers of 2. To get the parity bits, we see the bits related to it, eg: for parity bit 2, the bits related to it are D2, D3, D6, D7 and so on. Basically, bits in groups of 2 at gap of 2 bits are considered. This gives us the final hamming code to be sent to receiver. Hamming code for 4-bit data is: D7 D6 D5 P4 D3 P2 P1. It also gives us the exact position of error, in case of single bit errors.

**B.3: Conclusion:**

I have learnt the concept of error correction and its method, Hamming code. I have also successfully implemented Hamming code as a program.

**B.4: Questions of Curiosity:**

**If there is 1-Bit error. How can it be corrected on the receiver end using hamming code?**

Answer:

Code:

Text

Description automatically generated

Output:

Text

Description automatically generated

Text

Description automatically generated