



Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Spring, Year:2025), B.Sc. in CSE (Day)

Lab Report NO :03
Course Title: Data Communication Lab
Course Code: CSE 308/312 Section: 231_D1

Lab Experiment Name: Error Detection and Correction using Hamming Code in C.

Student Details

Name		ID
1.	Promod Chandra Das	231002005

Lab Date : 20-04-2025
Submission Date : 29-04-2025
Course Teacher's Name : Ms. Rusmita Halim Chaity

Lab Report Status

Marks:

Signature:.....

Comments:.....

Date:.....

1. TITLE OF THE LAB REPORT EXPERIMENT

Error Detection and Correction using Hamming Code in C .

2. OBJECTIVES

The objectives of this experiment are :

- To understand the concept and application of Hamming Code for single-bit error detection and correction.
- To implement Hamming(7,4) encoding and decoding using C programming.
- To simulate error detection and correction by entering a received code word.
- To identify the error position and correct it automatically.

3. PROCEDURE

1. Start by understanding the theory behind Hamming Codes, specifically the Hamming(7,4) scheme.
2. Design the logic to:
 - Calculate the number of parity bits required.
 - Insert data bits and initialize parity bit positions.
 - Compute parity bits using XOR operations.
3. Implement functions to:
 - Generate the Hamming code from user input.
 - Accept a received code word and compare it with the original.
 - Detect any single-bit error by recalculating parities.
 - Correct the erroneous bit if an error is detected.

4. IMPLEMENTATION

The C code used for this implementation is as follows:

```
#include <stdio.h>
#include <math.h>

int input[32];
int code[32];

int ham_calc(int pos, int n)
{ int i, parity = 0;
  pos = pos + 1;

  for (i = pos; i <= n; i++)
    { if ((i % pos) != 0) {
        parity ^= code[i - 1];
      }
    }
}
```

```

    return parity;
}
int main() {
    int n, i, p_n = 0, c_l, j = 0;

    printf("Please enter the length of the Data Word: "); scanf("%d",
    &n);

    printf("Please enter the Data Word:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &input[i]);
    }

    i = 0;
    while ((int)pow(2, i) < (n + i + 1))
        { p_n++;
          i++;
        }
    c_l = n + p_n;

    int m = 0;
    for (i = 1; i <= c_l; i++)
        { if ((i % 2) == 0)
          {
              code[i - 1] = 0;
          } else {
              code[i - 1] = input[m++];
          }
        }

    for (i = 0; i < p_n; i++) {
        int parity_pos = (int)pow(2, i) - 1;
        code[parity_pos] = ham_calc(parity_pos, c_l);
    }
    printf("The calculated Code Word is: ");
    for (i = 0; i < c_l; i++) {
        printf("%d", code[i]);
    }
    printf("\n");

    int received[32];
    printf("Please enter the received Code Word:\n");
    for (i = 0; i < c_l; i++) {
        scanf("%d", &received[i]);
    }

    int error_pos = 0;
    for (i = 0; i < p_n; i++) {
        int parity_pos = (int)pow(2, i);

```

```

    int parity = 0;
    for (j = 1; j <= c_l; j++) {
        if ((j % parity_pos) != 0)
            { parity ^= received[j - 1];
            }
        }
    if (parity != 0) {
        error_pos += parity_pos;
    }
}

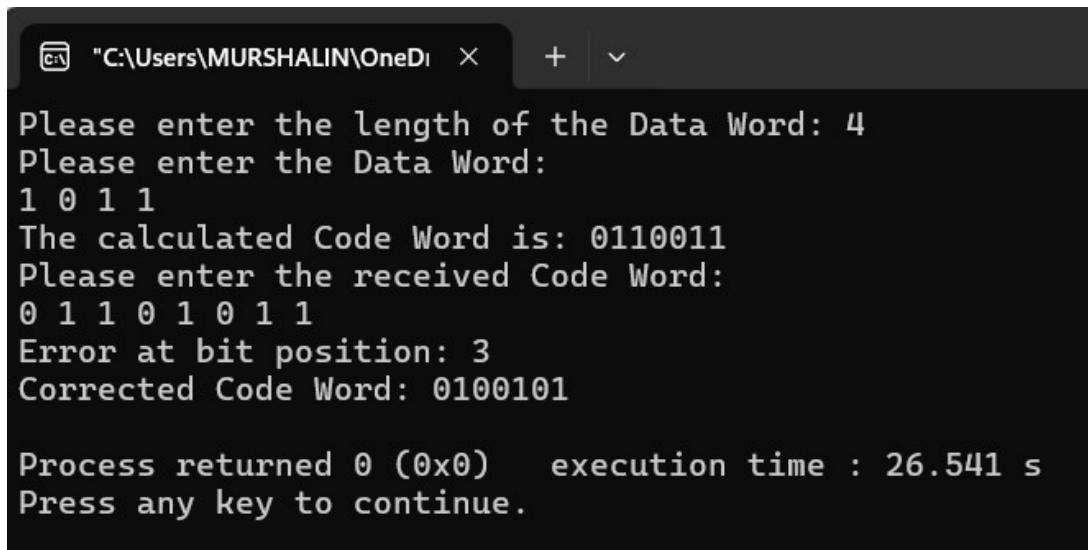
if (error_pos == 0) {
    printf("No error detected in the received code.\n");
} else {
    printf("Error at bit position: %d\n", error_pos);

    received[error_pos - 1] ^= 1;
    printf("Corrected Code Word: ");
    for (i = 0; i < c_l; i++) {
        printf("%d", received[i]);
    }
    printf("\n");
}

return 0;
}

```

5. TEST RESULT



```

C:\Users\MURSHALIN\OneD...
Please enter the length of the Data Word: 4
Please enter the Data Word:
1 0 1 1
The calculated Code Word is: 0110011
Please enter the received Code Word:
0 1 1 0 1 0 1 1
Error at bit position: 3
Corrected Code Word: 0100101

Process returned 0 (0x0)   execution time : 26.541 s
Press any key to continue.

```

Figure 1:Output

6. ANALYSIS AND DISCUSSION

1. Hamming Code is a reliable method for detecting and correcting single-bit errors in data transmission.
2. The parity bits are strategically placed at powers of 2 and calculated using XOR operations across specific bits.
3. The program successfully:
 - Calculated and embedded parity bits.
 - Simulated an error.
 - Detected and corrected the error by flipping the incorrect bit.
4. This lab demonstrates how mathematical logic and bitwise operations are used in real-world error control coding.

7. SUMMARY

In this lab, we implemented Hamming Code in C to detect and correct single-bit errors. We learned how to calculate parity bits, identify error positions using binary indexing, and correct errors in transmitted code. The code was validated with both correct and erroneous inputs, confirming its accuracy. This experiment provides practical knowledge on how error correction codes work in digital communication systems.