****

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Semester: (Spring, Year:2024), B.Sc. in CSE (Day)**

**Lab Report NO #02**

**Course Title: Data Communication Lab**

**Course Code: CSE 308 Section: 221 D20**

**Lab Experiment Name:** Implement of Hamming Code for error detection and correction approach using even parity.

**Student Details**

| **Name** | | **ID** |
| --- | --- | --- |
| **1.** | Tanvir Ahmed | 221002461 |

**Lab Date : 30/4/2024**

**Submission Date : 5/5/2024**

**Course Teacher’s Name : Md. Romazan Alom**

| **Lab Report Status**  **Marks: ………………………………… Signature:.....................**  **Comments:.............................................. Date:..............................** |
| --- |

**Introduction:**

Here we have a task, Hamming Code error direction and correction. We use even parity in the hamming code to encode the actual data for error detection and correction. That provides extra security that ensures no one can access our message without our permission and the main objective is to ensure we get the exact same data that we send. We are doing that by adding redundant bits in the corresponding position in the data. By doing that after receiving the data we can check those redundant bits and can say whether there is any error found or not.

Source code:

#**include** <stdio.h>

#**include** <math.h>

#**include** <string.h>

char main\_data[100];

char encoded\_data[100];

int **even\_parity\_Hamming**(int a, int b) {

int r = 0, i, j;

i = a - 1;

**while** (i < b) {

**for** (j = i; j < i + a; j++) {

**if** (encoded\_data[j] == '1')

r++;

}

i = i + 2 \* a;

}

**if** (r % 2 == 1)

**return** 1;

**else**

**return** 0;

}

int **main**() {

int i = 0, mlength = 0;

printf("Enter the Data in number: ");

scanf("%s", main\_data);

printf("\n");

mlength = strlen(main\_data);

int length, r = 0;

**while** (mlength > (int)pow(2, r) - (r + 1)) {

r++;

}

length = r + mlength;

int j, k;

j = k = 0;

**for** (i = 0; i < length; i++) {

**if** (i == ((int)pow(2, k) - 1)) {

encoded\_data[i] = '0';

k++;

} **else** {

encoded\_data[i] = main\_data[j];

j++;

}

}

**for** (i = 0; i < r; i++) {

int x = (int)pow(2, i);

int y = even\_parity\_Hamming(x, length);

**if**(y==0){

encoded\_data[x-1] = '0';

}

**else**{

encoded\_data[x-1] = '1';

}

}

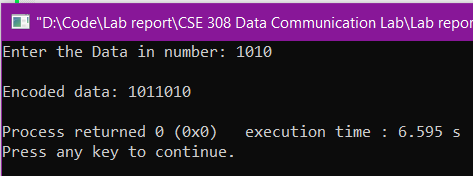
encoded\_data[length] = '\0';

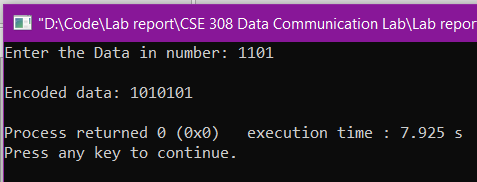
printf("Encoded data: %s\n", encoded\_data);

**return** 0;

}

output:





**Discussion:**

I have completed the task properly, and the output tested correctly. I took some help from the previous class learning and lab manual to complete the task. I have learned the hamming code error detection and correction using odd parity in the previous class. And I took that code, changed the odd parity to even parity based on my knowledge on parity bit. Finally, the output came properly.