EXPERIMENT NO. 5

SEMESTER: V **DATE OF PERFORMANCE: 08/08/22**

SUBJECT: CN Lab **DATE OF SUBMISSION: 13/08/22**

NAME OF THE STUDENT: Alston Fernandes **ROLL NO.: 16**

AIM	Write a program to simulate Hamming code generation, detection and correction.
LEARNING OBJECTIVE	The student will demonstrate the working of (7,4) hamming code.
LEARNING OUTCOME	The student will be able to detect and correct errors using hamming code.
COURSE OUTCOME	CSL502.3: Simulate and explore networking algorithms and protocols.
PROGRAM OUTCOME	PO1,PO2,PO3,PO4,PO5,PO9,PO10,PSO1,PSO2,PSO3
BLOOM'S TAXONOMY LEVEL	Apply
THEORY	**Parity bits:** The bit which is appended to the original data of binary bits so that the total number of 1s is even or odd. **Even parity:** To check for even parity, if the total number of 1s is even, then the value of the parity bit is 0. If the total number of 1s occurrences is odd, then the value of the parity bit is 1. **Odd Parity:** To check for odd parity, if the total number of 1s is even, then the value of parity bit is 1. If the total number of 1s is odd, then the value of parity bit is 0. **Algorithm of Hamming code:** • An information of 'd' bits are added to the redundant bits 'r' to form d+r. • The location of each of the (d+r) digits is assigned a decimal value. • The 'r' bits are placed in the positions 1,2,.....2^{k-1}. • At the receiving end, the parity bits are recalculated. The decimal value of the parity bits determines the position of an error.
LAB EXERCISE	1. Ask the user whether the program will work for even parity (or for odd)

	parity. 2. The user can enter the 4-bit data . 3. Complete Code Word for this and can be generated by calculating for the parity bits: P1=(D3,D5,D7) P2=(D3,D6,D7) P4=(D5,D6,D7) 4.Enter the Received codeword (with or without error).Notify the user to introduce error at only one bit position. 5.Check bits 1,3,5,7.....to generate C1.Check bits 2,3,6,7.....to generate C2.Check bits 4,5,6,7.....to generate C3.Generate the error codeword =C3C2C1. 6.If error is there,it will be reflected at which position and will display the corrected code word by inverting the respective bit. 7.Decode the data bits. /* Sample Output This is hamming code error detection and correction using EVEN parity Enter 4 data bits.D7 D6 D5 D3 Enter the value of D7:1 Enter the value of D6:0 Enter the value of D5:1 Enter the value of D3:0 3 parity bits are required for the transmission of data bits. SENDER: The data bits entered are: 1 0 1 0 The Parity bits are:

Value of P4 is 0

Value of P2 is 1

Value of P1 is 0

The Hamming code is as follows :-

D7 D6 D5 P4 D3 P2 P1

1 0 1 0 0 1 0

Enter the hamming code with error at any position of your choice.

NOTE: ENTER A SPACE AFTER EVERY BIT POSITION.

Error should be present only at one bit position

1 0 1 0 1 1 0

RECEIVER:

Error is detected at position 3 at the receiving end.

Correcting the error....

The correct code is 1 0 1 0 0 1 0

The decoded data is:1010

*/

Code:

```
# include <stdio.h>

# include <stdlib.h>

# include <string.h>

int PARITY;
```

```
int checkCW(char* code) {
        if ((*(code + 6) +*(code + 4) + *(code + 2) + *(code + 0)) % 2 !=
PARITY)
                return 0;
        if ((*(code + 5) +*(code + 4) + *(code + 1) + *(code + 0)) % 2 !=
PARITY)
                return 0;
        if ((*(code + 3) +*(code + 2) + *(code + 1) + *(code + 0)) % 2 !=
PARITY)
                return 0;
        return 1;
}

char* correctCW(char* code) {
        int c[3] = {0};
        if ((*(code + 6) + *(code + 4) + *(code + 2) + *(code + 0)) % 2 !=
PARITY)
                c[0]++; // C1
        if ((*(code + 5) + *(code + 4) + *(code + 1) + *(code + 0)) % 2 !=
PARITY)
                c[1]++; // C2
```

```
        if ((*(code + 3) + *(code + 2) + *(code + 1) + *(code + 0)) % 2 !=
PARITY)

                c[2]++; // C3

        printf("\nError code generated (C3 C2 C1) is: %d %d %d", c[2],
c[1], c[0]);

        int index = (2 * 2) * c[2] + (2 * 1) * c[1] + (1) * c[0];

        printf("\nError detected at position %d at recieveing end",index);

        printf("\nCorrecting the error...\n");

        *(code + 7 - index) = *(code +  7 - index) ? '0' : '1';

        return code;

}



int main() {

        char choice;

        char cw[7];

        char data[4];

        printf("Even or Odd parity(e/o): ");

        scanf("\n %c",&choice);

        if (choice == 'e' || choice == 'E')

                PARITY = 0;
```

```
        else if (choice == 'o' || choice == 'O')
                PARITY = 1;
        else {
                printf("Wrong choice");
                return 0;
        }

        printf("\nEnter the recieved 4-bit data in binary format(D7 D6 D5 D3): ");
        for (int i = 0; i < 4; i++)
        {
                scanf("\n %c",&data[i]);
                if (data[i] != '0' && data[i] != '1')
                {
                        printf("\nError! Wrong Format\n");
                        return 0;
                }
        }
        printf("\n3 Parity bits need to be generated\n");
        printf("\n-------SENDER-------\n");
```

```
        cw[0] = data[0];

        cw[1] = data[1];

        cw[2] = data[2];

        cw[3] = ((data[0] + data[1] + data[2]) % 2 == PARITY) ? '0' : '1';

        cw[4] = data[3];

        cw[5] = ((data[0] + data[1] + data[3]) % 2 == PARITY) ? '0' : '1';

        cw[6] = ((data[0] + data[2] + data[3]) % 2 == PARITY) ? '0' : '1';

        printf("\nData bits:\n");

        for (int i = 0; i < 4; i++)

        {

                printf("%c ",data[i]);

        }

        printf("\nParity bits generated(P1 P2 P4): %c %c
%c",cw[6],cw[5],cw[3]);




        printf("\nTransmitted bit stream(D7 D6 D5 P4 D3 P2 P1): ");

        for (int i = 0; i < 7; i++)

        {

                printf("%c ",cw[i]);
```

```
        }



        printf("\nOnly 1 bit error is permitted\n");

        printf("Enter the recieved 7-bit codeword in binary format(MSB-
LSB): ");

        for (int i = 0; i < 7; i++)

        {

                scanf("\n %c",&cw[i]);

                if (cw[i] != '0' && cw[i] != '1')

                {

                        printf("Error!\n");

                        return 0;

                }

        }

        printf("\n-------RECIEVER-------\n");

        printf("\nRecieved codeword is: ");

        for (int i = 0; i < 7; i++)

        {

                printf("%c ", cw[i]);

        }
```

```
        if (!checkCW(cw))
        {
                correctCW(cw);
        }
        printf("\nCorrect codeword is ");
        for (int i = 0; i < 7; i++)
        {
                printf("%c ", cw[i]);
        }
        printf("\nDecoded data is\n%c %c %c %c\n", cw[0], cw[1], cw[2],
cw[4]);
        return 0;
}
```

Even Parity 1 bit error:

```
alston@acer ~/Projects/sem5 $ gcc hamming.c -o hamming && ./hamming
Even or Odd parity(e/o): e

Enter the recieved 4-bit data in binary format(D7 D6 D5 D3): 1 0 1 0

3 Parity bits need to be generated

-------SENDER-------

Data bits:
1 0 1 0
Parity bits generated(P1 P2 P4): 0 1 0
Transmitted bit stream(D7 D6 D5 P4 D3 P2 P1): 1 0 1 0 0 1 0
Only 1 bit error is permitted
Enter the recieved 7-bit codeword in binary format(MSB-LSB):  1 0 1 0 1 1 0

-------RECIEVER-------

Recieved codeword is: 1 0 1 0 1 1 0
Error code generated (C3 C2 C1) is: 0 1 1
Error detected at position 3 at recieveing end
Correcting the error...

Correct codeword is 1 0 1 0 0 1 0
Decoded data is
1 0 1 0
alston@acer ~/Projects/sem5 $
```

Even Parity no error:

```
alston@acer ~/Projects/sem5 $ gcc hamming.c -o hamming && ./hamming
Even or Odd parity(e/o): e

Enter the recieved 4-bit data in binary format(D7 D6 D5 D3): 1 0 1 1

3 Parity bits need to be generated

-------SENDER-------

Data bits:
1 0 1 1
Parity bits generated(P1 P2 P4): 1 0 0
Transmitted bit stream(D7 D6 D5 P4 D3 P2 P1): 1 0 1 0 1 0 1
Only 1 bit error is permitted
Enter the recieved 7-bit codeword in binary format(MSB-LSB): 1 0 1 0 1 0 1

-------RECIEVER-------

Recieved codeword is: 1 0 1 0 1 0 1
Correct codeword is 1 0 1 0 1 0 1
Decoded data is
1 0 1 1
alston@acer ~/Projects/sem5 $
```

Odd Parity 1 bit error:

```
alston@acer ~/Projects/sem5 $ gcc hamming.c -o hamming && ./hamming
Even or Odd parity(e/o): o

Enter the recieved 4-bit data in binary format(D7 D6 D5 D3): 1 1 0 0

3 Parity bits need to be generated

-------SENDER-------

Data bits:
1 1 0 0
Parity bits generated(P1 P2 P4): 0 1 1
Transmitted bit stream(D7 D6 D5 P4 D3 P2 P1): 1 1 0 1 0 1 0
Only 1 bit error is permitted
Enter the recieved 7-bit codeword in binary format(MSB-LSB): 1 1 1 1 0 1 0

-------RECIEVER-------

Recieved codeword is: 1 1 1 1 0 1 0
Error code generated (C3 C2 C1) is: 1 0 1
Error detected at position 5 at recieveing end
Correcting the error...

Correct codeword is 1 1 0 1 0 1 0
Decoded data is
1 1 0 0
alston@acer ~/Projects/sem5 $
```

Odd Parity no error:

```
alston@acer ~/Projects/sem5 $ gcc hamming.c -o hamming && ./hamming
Even or Odd parity(e/o): o

Enter the recieved 4-bit data in binary format(D7 D6 D5 D3): 1 0 0 1

3 Parity bits need to be generated

-------SENDER-------

Data bits:
1 0 0 1
Parity bits generated(P1 P2 P4): 1 1 0
Transmitted bit stream(D7 D6 D5 P4 D3 P2 P1): 1 0 0 0 1 1 1
Only 1 bit error is permitted
Enter the recieved 7-bit codeword in binary format(MSB-LSB): 1 0 0 0 1 1 1

-------RECIEVER-------

Recieved codeword is: 1 0 0 0 1 1 1
Correct codeword is 1 0 0 0 1 1 1
Decoded data is
1 0 0 1
alston@acer ~/Projects/sem5 $
```

Even Parity 2 bit error (Wrong answer):

```
alston@acer ~/Projects/sem5 $ gcc hamming.c -o hamming && ./hamming
Even or Odd parity(e/o): e

Enter the recieved 4-bit data in binary format(D7 D6 D5 D3): 1 0 1 0

3 Parity bits need to be generated

-------SENDER-------

Data bits:
1 0 1 0
Parity bits generated(P1 P2 P4): 0 1 0
Transmitted bit stream(D7 D6 D5 P4 D3 P2 P1): 1 0 1 0 0 1 0
Only 1 bit error is permitted
Enter the recieved 7-bit codeword in binary format(MSB-LSB): 1 0 1 0 1 0 0

-------RECIEVER-------

Recieved codeword is: 1 0 1 0 1 0 0
Error code generated (C3 C2 C1) is: 0 0 1
Error detected at position 1 at recieveing end
Correcting the error...

Correct codeword is 1 0 1 0 1 0 0
Decoded data is
1 0 1 1
alston@acer ~/Projects/sem5 $
```

Non binary input error:

```
alston@acer ~/Projects/sem5 $ gcc hamming.c -o hamming && ./hamming
Even or Odd parity(e/o): e

Enter the recieved 4-bit data in binary format(D7 D6 D5 D3): 1 0 1 a

Error! Wrong Format
alston@acer ~/Projects/sem5 $ gcc hamming.c -o hamming && ./hamming
Even or Odd parity(e/o): o

Enter the recieved 4-bit data in binary format(D7 D6 D5 D3): 2 1 0 1

Error! Wrong Format
alston@acer ~/Projects/sem5 $
```

REFERENCES	• B.A. Forouzan, "Data Communications and Networking", TMH, Fourth Edition. • https://www.javatpoint.com/computer-network-error-correction