

<u>NAMES</u>	<u>ROLL NO.'S</u>
<u>AMEEMA ARIF</u>	<u>CS-17122</u>
<u>ABDULLAH ANSARI</u>	<u>CS-17080</u>

YEAR : THIRD YEAR

BATCH : 2017

COURSE : DIGITAL COMMUNICATION SYSTEMS

COURSE CODE :CS-352

**PROJECT TITLE : HAMMING BASED ERROR
DETECTION & CORRECTION**

SUBMITTED TO : SIR UMER IFTIKHAR

PROJECT REPORT

OBJECTIVE :

It is a program which will calculate and add parity bits accordingly for transferring a maximum 16 bit binary code, in order to detect and correct a single bit error in the code on the receiving end , which was occurred during transmission.

BASIC TERMS & CONCEPTS :

- **Hamming Code:**

Hamming code is an Error Correction code which can be used to detect and correct bit errors that can occur when computer data is moved or stored.

- **Parity Bits :**

Parity bits are extra binary digits that are generated and transferred along with a data transfer to ensure that no bits were lost during a data transfer.

- **Error :**

A measure of the estimated difference between the observed value of a quantity and its true value.

There are three types of error :

- (i) Single bit error
- (ii) Multi bits error
- (iii) Burst bits error

- **Error Detection :**

Detection of errors which are caused by noise or other impairments during transmission from the sender to the receiver.

- **Error Correction :**

It is the detection of errors and re-construction of the original signal by some method. This will produce error-free data.

WORKING :

Calculation of Parity Bits :

We are allowing the user to enter maximum 16 bits. Which means, on a maximum we can have 5 parity bits.

Formula to calculate the no. of parity bits : $2^r \geq n+r+1$

Where,

r : no. of parity bits

n : no. of bits given as input

These parity bits will be represented in the code on the following positions :

$R_1 : 2^0$; $R_2 : 2^1$; $R_3 : 2^2$; $R_4 : 2^3$; $R_5 : 2^4$

R_1 will take care of these bits

D	D	D	D	D	R	D	D	D	D	D	D	D	R	D	D	D	R	D	R	R
16	15	14	13	12	5	11	10	9	8	7	6	5	4	4	3	2	3	1	2	1

R_2 will take care of these bits

D	D	D	D	D	R	D	D	D	D	D	D	R	D	D	D	R	D	R	R
16	15	14	13	12	5	11	10	9	8	7	6	5	4	4	3	2	3	1	2

R_3 will take care of these bits

D	D	D	D	D	R	D	D	D	D	D	D	R	D	D	D	R	D	R	R	
16	15	14	13	12	5	11	10	9	8	7	6	5	4	4	3	2	3	1	2	1

R_4 will take care of these bits

D	D	D	D	D	R	D	D	D	D	D	D	R	D	D	D	R	D	R	R	
16	15	14	13	12	5	11	10	9	8	7	6	5	4	4	3	2	3	1	2	1

R₅ will take care of these bits

D	D	D	D	D	R	D	D	D	D	D	D	D	R	D	D	D	R	D	R	R
16	15	14	13	12	5	11	10	9	8	7	6	5	4	4	3	2	3	1	2	1

Now the bits which are taken care by each R will be XORed together to find the respective parity bit. Note that while XORing the bits we will not take parity bits as they are not defined yet.

Addition of Error in the Code :

To test the functioning of our hamming code we have added an error in the code which has to be transferred. First, we generated a random bit position which has to be effected by the error. Our code will generate a different bit position every time when we will execute our code. The bit on the generated position will be inverted in order to create an error.

Decoding of the Received Code :

For decoding, first we will again calculate the parity bits in accordance with our received code using the same method as done previously. Then we will compare the calculated parity bits to the bits present in our received code on the respective parity bit position.

The positions of the calculated parity bits which are different from the ones present in our received code will be added together and their sum will indicate the position of bit which is in error.

Error Correction :

To correct the error we will simply invert that bit which was detected as the error bit.

Final Code :

Finally, we will remove the parity bits from the corrected code and get the exactly same code back which was transmitted.

SAMPLE OUTPUTS

- When input is given :

```
Command Window

Insert a Binary String of max 16 bits between square Brackets [] :[1 0 1 0]
Number of parity bits for the given input code =
    3

Parity bit 1 = 1
Parity bit 2 = 0
Parity bit 3 = 1
After inserting Parity bits we got the following code :
    1    0    1    1    0    1    0

Code with noise
    1    0    1    0    0    1    0

Parity bit 1 = 1
Parity bit 2 = 0
Parity bit 3 = 1
Error found on the following bit no. :
    4

Code after error removal :
    1    0    1    1    0    1    0

Detected Error has been corrected
Code corrected sucessfully
The code given by User :
    1    0    1    0

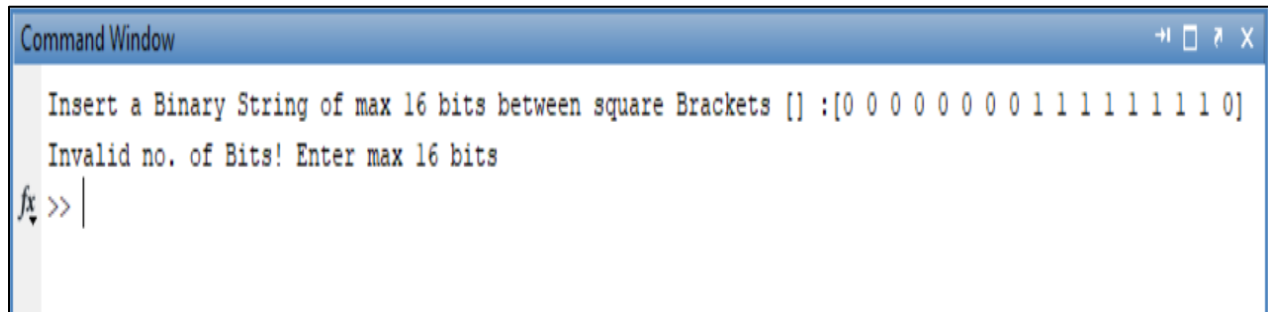
The Final received code is :
    1    0    1    0
```

- When no input is given :

```
Command Window

Insert a Binary String of max 16 bits between square Brackets [] :[]
No input given
fx >> |
```

- **When more than 16 bits are given as input :**



```
Command Window
Insert a Binary String of max 16 bits between square Brackets [] :[0 0 0 0 0 0 0 1 1 1 1 1 1 1 0]
Invalid no. of Bits! Enter max 16 bits
fx >> |
```

APPLICATIONS OF HAMMING CODE :

Here are some common applications of using Hamming code:

- Satellites
- Computer Memory
- Modems
- Plasma CAM
- Open connectors
- Shielding wire
- Embedded Processor

ADVANTAGES OF HAMMING CODE :

- Hamming code method is effective on networks where the data streams are given for the single-bit errors.
- Hamming code not only provides the detection of a bit error but also helps you to indent bit containing error so that it can be corrected.
- The ease of use of hamming codes makes it best them suitable for use in computer memory and single-error correction.

