

# **Data Communication & Computer Networks Lab**

**Submitted by: Submitted to:**

**NAME : Rahul Gusain**

**SAP : 500084143**

**ROLL NO. : R214220900 Ved Prakash Bhardwaj**

**BATCH: 1 DevOps**

**EXPERIMENT 4**

Aim :- To implement hamming code.

**Theory:-**

Hamming code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction

**Algorithm:-**

1.Number the bits starting from 1: bit 1, 2, 3, 4, 5, 6, 7, etc.

2.Write the bit numbers in binary: 1, 10, 11, 100, 101, 110, 111, etc.

3.All bit positions that are powers of two (have a single 1 bit in the binary form of their position) are parity bits: 1, 2, 4, 8, etc. (1, 10, 100, 1000)

4.All other bit positions, with two or more 1 bits in the binary form of their position, are data bits.

5.Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.

* 1. Parity bit 1 covers all bit positions which have the **least** significant bit set: bit 1 (the parity bit itself), 3, 5, 7, 9, etc.
  2. Parity bit 2 covers all bit positions which have the **second** least significant bit set: bits 2-3, 6-7, 10-11, etc.
  3. Parity bit 4 covers all bit positions which have the **third** least significant bit set: bits 4–7, 12–15, 20–23, etc.
  4. Parity bit 8 covers all bit positions which have the **fourth** least significant bit set: bits 8–15, 24–31, 40–47, etc.
  5. In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

If a byte of data to be encoded is 10011010, then the data word (using \_ to represent the parity bits) would be \_\_1\_001\_1010, and the code word is 011100101010.

The choice of the parity, even or odd, is irrelevant but the same choice must be used for both encoding and decoding.

**Code:-**

#include<stdio.h>

void main() {

int data[10];

int dataatrec[10],c,c1,c2,c3,i;

printf("Enter 4 bits of data one by one\n");

scanf("%d",&data[0]);

scanf("%d",&data[1]);

scanf("%d",&data[2]);

scanf("%d",&data[4]);

//Calculation of even parity

data[6]=data[0]^data[2]^data[4];

data[5]=data[0]^data[1]^data[4];

data[3]=data[0]^data[1]^data[2];

printf("\nEncoded data is\n");

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

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

printf("\n\nEnter received data bits one by one\n");

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

scanf("%d",&dataatrec[i]);

c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];

c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];

c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];

c=c3\*4+c2\*2+c1 ;

if(c==0) {

printf("\nNo error while transmission of data\n");

}

else {

printf("\nError on position %d",c);

printf("\nData sent : ");

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

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

printf("\nData received : ");

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

printf("%d",dataatrec[i]);

printf("\nCorrect message is\n");

//if errorneous bit is 0 we complement it else vice versa

if(dataatrec[7-c]==0)

dataatrec[7-c]=1;

else

dataatrec[7-c]=0;

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

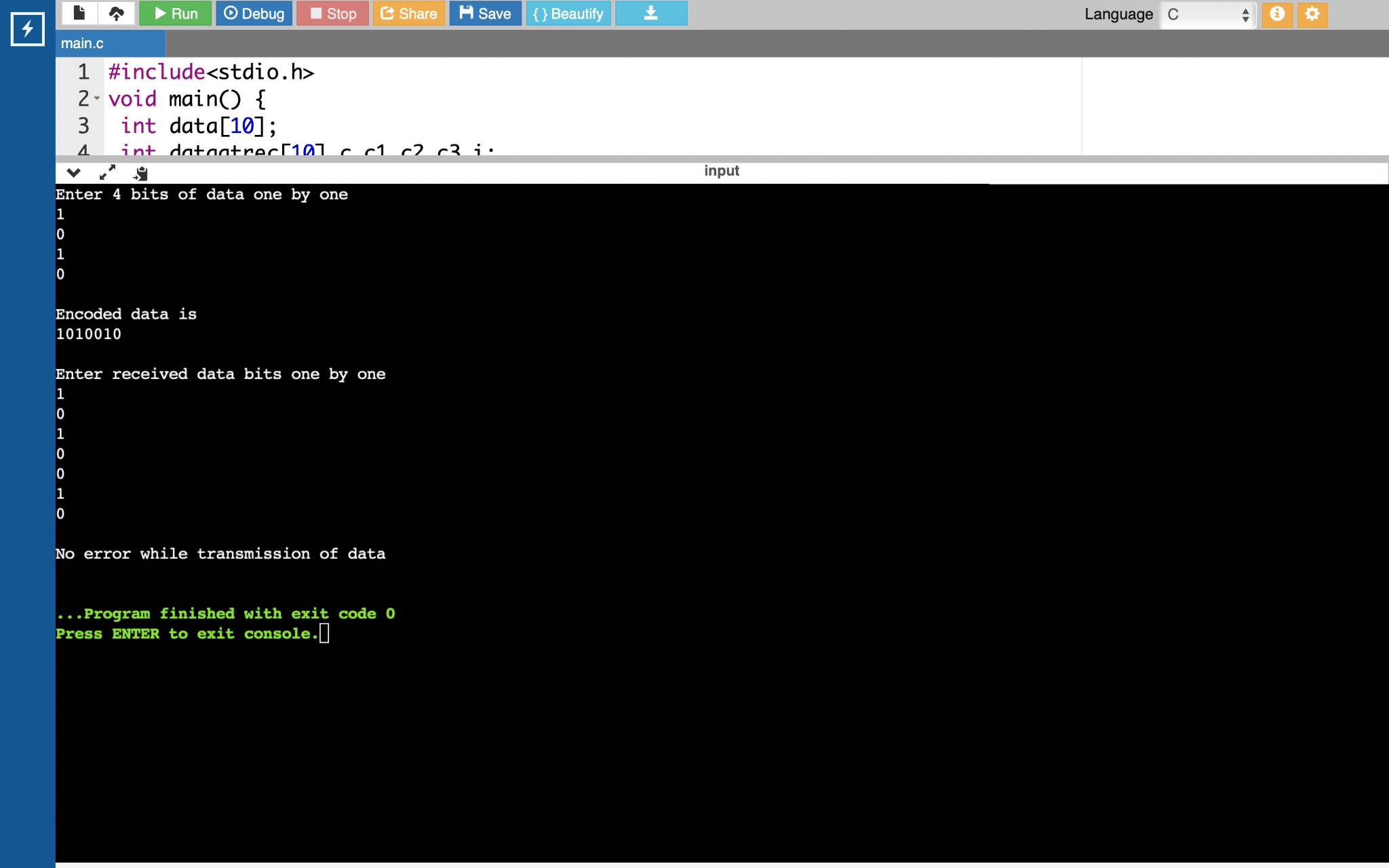
printf("%d",dataatrec[i]);

}

}

}

**Output:-**

****