

Experiment No. 5

Aim: Hamming code Implementation for Error detection.

Theory:

what is hamming code?

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 technique developed by R.W. Hamming for error correction.

Redundant bits

Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer.

The number of redundant bits can be calculated using the following formula:

$$2^r \geq m + r + 1$$

where, r = redundant bit, m = data bit

Suppose the number of data bits is 7, then the number of redundant bits can be calculated using:

$$= 2^4 \geq 7 + 4 + 1$$

Thus, the number of redundant bits = 4

Parity bits

A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data are even or odd. Parity bits are used for error detection. There are two types of parity bits:

1. Even parity bit:

In the case of even parity, for a given set of bits, the number of 1's are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1's an even number. If the total number of 1's in a given set of bits is already even, the parity bit's value is 0.

2. Odd Parity bit :

In the case of odd parity, for a given set of bits, the number of 1's are counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1's an odd number. If the total number of 1's in a given set of bits is already odd, the parity bit's value is 0.

Calculation hamming code :

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8..)
2. All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7..)

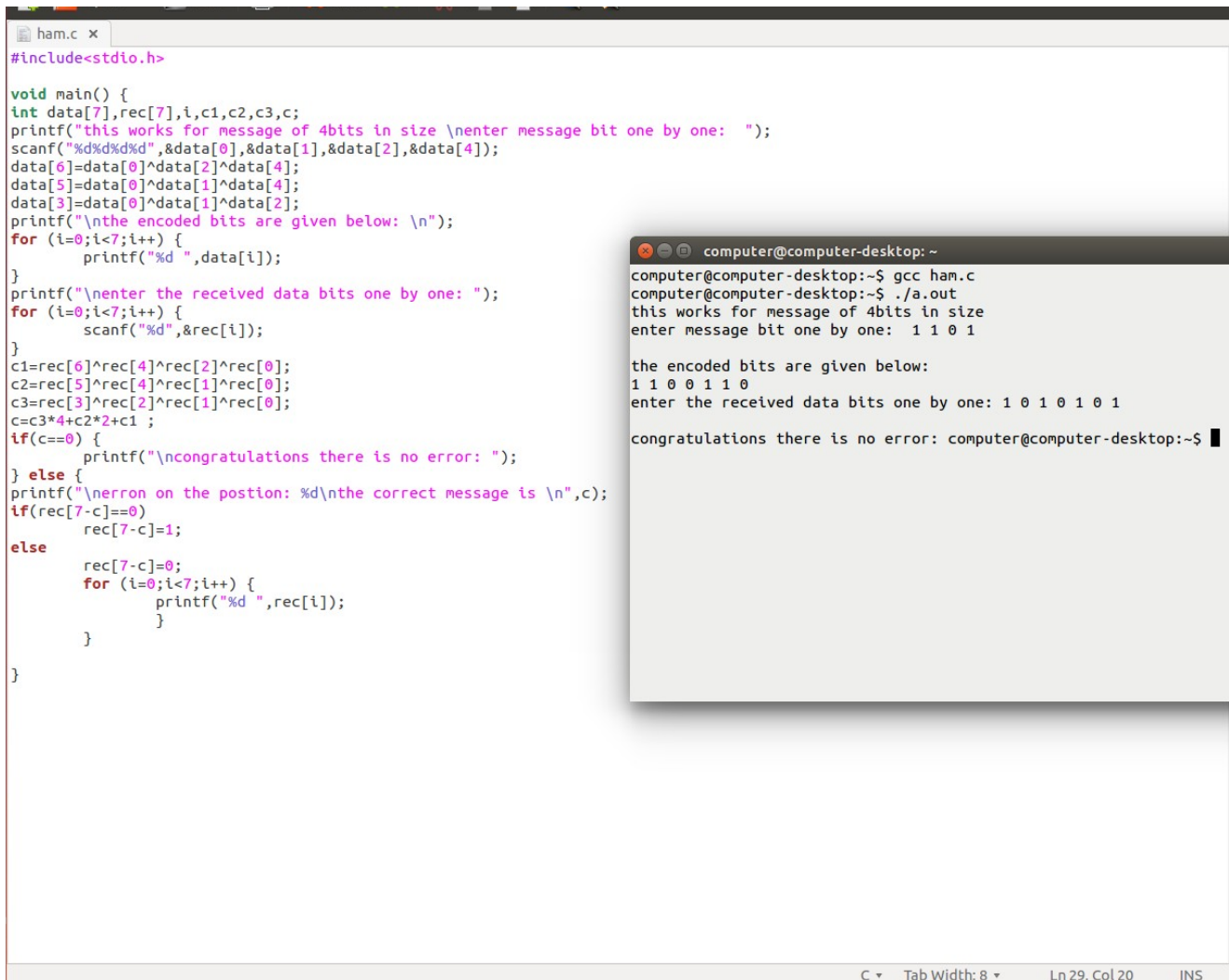
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.

Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,...)

Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,...)

Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7..)

4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.



The image shows a C program named `ham.c` in a text editor and its execution in a terminal. The program implements a Hamming code for 4 data bits, using 3 parity bits (positions 1, 2, and 4). The code calculates the parity bits and then checks for errors in the received data. If no error is found, it prints a congratulatory message.

```
#include<stdio.h>

void main() {
    int data[7],rec[7],i,c1,c2,c3,c;
    printf("this works for message of 4bits in size \nenter message bit one by one: ");
    scanf("%d%d%d%d",&data[0],&data[1],&data[2],&data[4]);
    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("\nthe encoded bits are given below: \n");
    for (i=0;i<7;i++) {
        printf("%d ",data[i]);
    }
    printf("\nenter the received data bits one by one: ");
    for (i=0;i<7;i++) {
        scanf("%d",&rec[i]);
    }
    c1=rec[6]^rec[4]^rec[2]^rec[0];
    c2=rec[5]^rec[4]^rec[1]^rec[0];
    c3=rec[3]^rec[2]^rec[1]^rec[0];
    c=c3*4+c2*2+c1;
    if(c==0) {
        printf("\ncongratulations there is no error: ");
    } else {
        printf("\nerron on the postion: %d\nthe correct message is \n",c);
        if(rec[7-c]==0)
            rec[7-c]=1;
        else
            rec[7-c]=0;
        for (i=0;i<7;i++) {
            printf("%d ",rec[i]);
        }
    }
}
```

Terminal output:

```
computer@computer-desktop: ~
computer@computer-desktop:~$ gcc ham.c
computer@computer-desktop:~$ ./a.out
this works for message of 4bits in size
enter message bit one by one: 1 1 0 1

the encoded bits are given below:
1 1 0 0 1 1 0
enter the received data bits one by one: 1 0 1 0 1 0 1

congratulations there is no error: computer@computer-desktop:~$
```

Conclusion: Hence the hamming code progreem is successfully executed.