Output (checksum) :

MODIFIED

```
$ ./ checksum
4500  0073  0000  4000  4011  b861  c0a8 0001
 c0a8   00C7
 Computed checksum : b861
 Enter 0 to modify, anything else to not modify
 0
 Invalid checksum
```

UNMODIFIED

```
$ ./ checksum
4500 0073 0000 4000 4011 b861 c0a8 0001
 c0a8   00c7                          (entered)
 Computed checksum : b861               1
 Valid checksum
 $
```

3. Write programs to implement error detection and correction concept using Checksum and Hamming code.

```
// Checksum
#include <stdio.h>
#include <stdlib.h>

unsigned int checksumGenerate (uint16_t * ipHeader)
{
    unsigned int sum = 0;
    ipHeader[5] = 0;
    for(int i=0; i<10; i++)
    {
        sum += ipHeader[i];
        while ((sum >> 16))
            sum = (sum & 0xFFFF) + 1u;
    }
    ipHeader[5] = (sum & 0xFFFF);
    ipHeader[5] = ~ipHeader[5];
    return ipHeader[5];
}

int verifyChecksum (uint16_t * ipHeader)
{
    unsigned int sum = 0u;

    //continued.
```

R.V. COLLEGE OF ENGINEERING

OBSERVATION / DATA SHEET

Date _____ Name M. C. SOHAN.

Dept./Lab _____ Class _____ Expt./No. _____

Title Checksum

```
for(int i=0; i<10; i++)
{
    sum += ipHeader[i];
    while (sum>>16)
        sum = (sum & 0xFFFF) + 1u;
}
if ((~sum & 0xFFFF) == 0)
    return 1;
else return 0;
}
int main()
{
    uint16_t ipfield[10];
    unsigned int sum, modify;
    for(int i=0; i<10; i++)
        scanf("%x", (ipfield+i));
    sum = checksumGenerate(ipfield) & 0xffff;
```

Signature of
Teacher incharge

```
printf ("Computed checksum: %x", sum);

printf ("\n Enter 0 to not modify, anything else
        to modify");
scanf ("%u", modify);
if (u)
    ipfield[0] = 0x0011 ^ ipfield[0];
// this changed (toggled) the position values

if (verifyChecksum (ipfield))
    printf ("Valid checksum");
else
    printf ("Invalid checksum");

return 0;
} // end of program.
```

Date

Expt. No. 3 Page No. 7

```
// Hamming code

#include <stdio.h>

int main()
{
    int bit[8];
    int rcvbit[8];
    int s, s0, s1, s2, i;
    printf(" enter 4 bits of data \n");
    for (i=0; i<4; i++)
        scanf("%d", &bit[i]);
// printf("Data entered!");
// calculate the values for the various bit positions
    bit[6] = bit[0] ^ bit[2] ^ bit[4];
    bit[5] = bit[0] ^ bit[1] ^ bit[4];
    bit[3] = bit[0] ^ bit[1] ^ bit[2];

printf("Data Encoded : \n");
printf(" d3 d2 d1 r2 d0 r1 r0 \n");
for( i=0; i<7; i++ )
        printf("%d", bit[i]);

printf(" Input received bits : \n");
    for(i=0; i<7; i++)
        scanf("%d", &rcvbit[i]);

// cont..
```

Teacher's Signature : _____

Output (Hamming Code):

```
$ ./hammingcode
Enter 4 bit data:
1 0 0 0
Encoded bits:
d3 d2 d1 r2 d0 r1 r0
1  0  0  1  0  1  1
Input received bits
1 0 0 0 0 1 1
Received sequence
d3 d2 d1 r2 d0 r1 r0
1  0  0  0  0  1  1
Error at position 4
Corrected message: 1001011
```

One complete execution

```
$ ./hamming code
```

2nd execution input

input given: 0110
for received bits: 0110011

```
Encoded bits:
d3 d2 d1 r2 d0 r1 r0
0  1  1  0  0  1  1

No error Detected.
```

Second execution Output

```
// calculate syndrome
s0 = rcvbit[6] ^ rcvbit[4] ^ rcvbit[2] ^ rcvbit[0];
s1 = rcvbit[5] ^ rcvbit[4] ^ rcvbit[1] ^ rcvbit[0];
s2 = rcvbit[3] ^ rcvbit[2] ^ rcvbit[1] ^ rcvbit[0];
s = (s2<<2) + (s1<<1) + s0;
if (s>0)
    printf ("No error detected");
else {
    printf ("\n Recieved sequence \n");
    printf (" d3 d2 d1 r2 d0 r1 r0\n");
    for( i=0; i<7 ; i++)
        printf("%d", rcvbit[i]);
    printf ("Error in position :");
    printf ("%d", s);
    if ( rcvbit[7-s] ==0)
        rcvbit[7-s] = 1;
    else
        rcvbit [7-s] = 0;
    for( i=0; i<7; i++)
        printf ("%d ", rcvbit[i]);

    }
printf ("\n");
return 0;
}   // end of program.
```