

BCSE308 P -COMPUTER NETWORKS LAB

Register Number	24BCE1282	Subject Code	BCSE308P
Name of the Student	Kunsh Jain	Subject Name	Computer Networks Lab
Programme	B.Tech		
Date	23/7/25	Exp No	4

Aim :

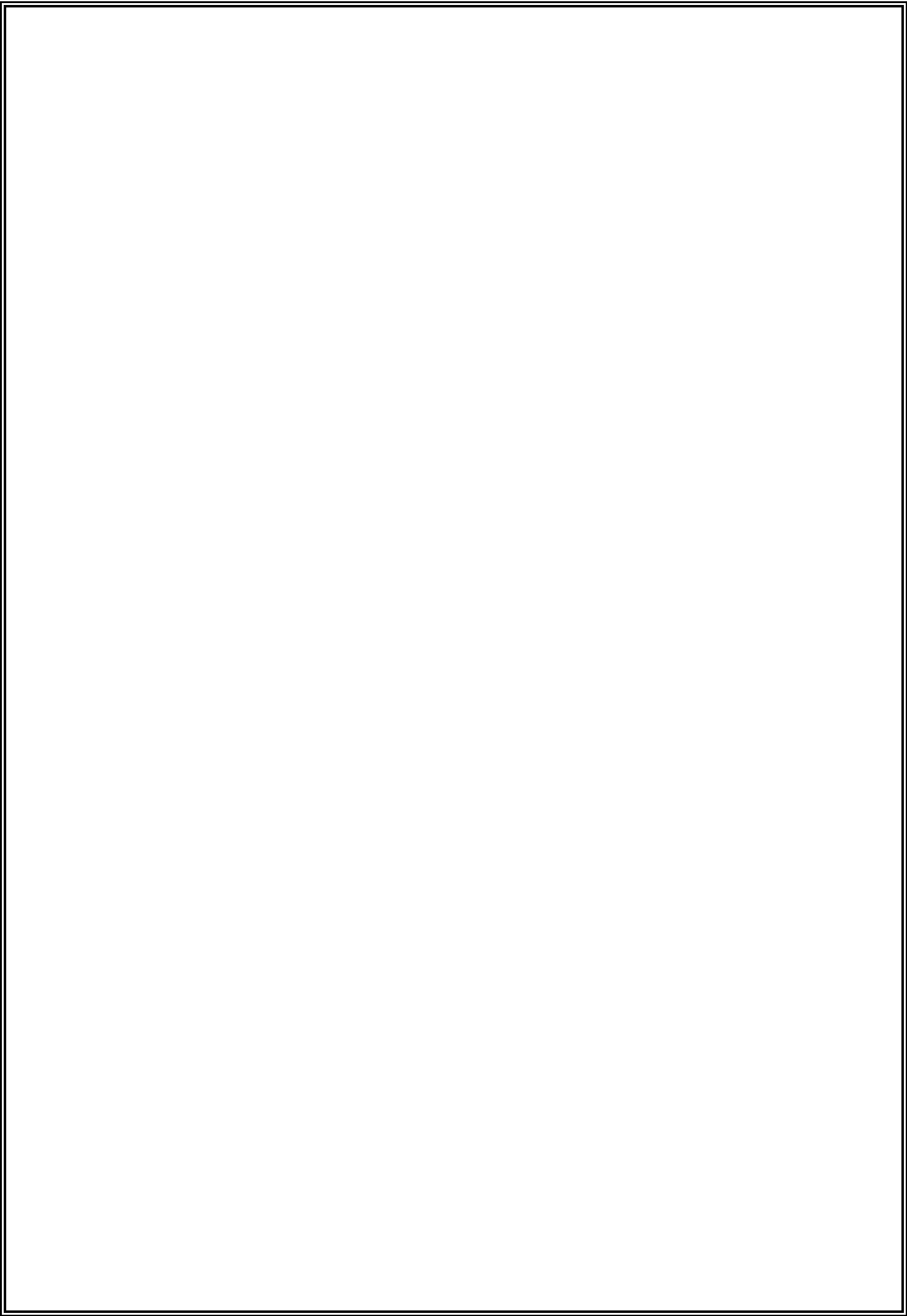
(a) To write a c/cpp program to generate CRC lists for the given divisor is 4 bits+message is 5 bits.

(b) Generate checksum bits of 8 bits for 5 sections.

Procedure:

1. Write the formula for the assumed calculations
2. Do dry calculations on your notebook
3. Check the output with the verified output
4. Write the c/cpp code for (a) and (b)
5. Get the output verified

Attestation:



LAB-4

Page No.:
Date: / /

- 1) Write a C++ program to generate CRC bits for a given divisor 4 bits and message is 5 bits
- 2) Generate checksum bits for 8 bit for 5 sections

1(a) Assuming bits Message = 5
Divisor = 4
Remainder = 101

10011
10101000
1011
000100
1011
01110
1011
0101

1 10101001 169
2 11001100 204
3 11110000 240
4 00011110 30
5 10111001 185
0 828

1) 10101001 + 11001100 = 10110101
2) 101110101 + 11110000 = 1001100101

3) 1001100101 + 00011110 = 1010000011
4) 1010000011 + 10111001 = 100111100

0000000
10101001
11001100
11110000
00011110
+ 10111001
1111100

0011100
+ 11
0011111

11000000

checksum = 11000000

(a) OUTPUT:

CRC BIT - 101

CODEWORD - 10101101

for calc P.T.O

(~) Bit assumed = 1010

Divisor = 1011

CRC Bit generated = 101

(b) Bits:

1	169	10101001
2	204	11001100
3	240	11100000
4	30	0001110
5	185	10111001

total sum = 1100111100 = 828

After c added = 00111111

After 1's comp = 11000000

Hence Checksum = 11000000

(192)

OUTPUT:

(a) CRC bits: 101

Codeword: 10101101

o/p verified
M.S.R.

(b) Enter 5 data values (each 0-255)

169

204

240

30

185

Checksum (8-bit) = 192

8 bit res = 11000000

o/p verified
M.S.R.

Code:

(a)

```

1  #include <stdio.h>
2  #define DATA_BITS 5
3  #define DIV_BITS 4
4
5  int main() {
6      int data[DATA_BITS + DIV_BITS - 1];
7      int divisor[DIV_BITS];
8      int temp[DATA_BITS + DIV_BITS - 1];
9      int i, j;
10     int dataword[DATA_BITS] = {1, 0, 1, 0, 1};
11     int divisor_input[DIV_BITS] = {1, 0, 1, 1};
12
13     for (i = 0; i < DATA_BITS; i++)
14         data[i] = dataword[i];
15     for (i = DATA_BITS; i < DATA_BITS + DIV_BITS - 1; i++)
16         data[i] = 0;
17
18     for (i = 0; i < DATA_BITS + DIV_BITS - 1; i++)
19         temp[i] = data[i];
20     for (i = 0; i < DIV_BITS; i++)
21         divisor[i] = divisor_input[i];
22
23     for (i = 0; i < DATA_BITS; i++) {
24         if (temp[i] == 1) {
25             for (j = 0; j < DIV_BITS; j++)
26                 temp[i + j] ^= divisor[j];
27         }
28     }
29
30     printf("CRC bits: ");
31     for (i = DATA_BITS; i < DATA_BITS + DIV_BITS - 1; i++)
32         printf("%d", temp[i]);
33     printf("\nCodeword: ");
34     for (i = 0; i < DATA_BITS; i++)
35         printf("%d", dataword[i]);
36     for (i = DATA_BITS; i < DATA_BITS + DIV_BITS - 1; i++)
37         printf("%d", temp[i]);
38     printf("\n");
39
40     return 0;
41 }
42
43

```

#include <stdio.h>

#define DATA_BITS 5

#define DIV_BITS 4

int main() {

int data[DATA_BITS + DIV_BITS - 1];

int divisor[DIV_BITS];

```

int temp[DATA_BITS + DIV_BITS - 1];

int i, j;

int dataword[DATA_BITS] = {1, 0, 1, 0, 1};

int divisor_input[DIV_BITS] = {1, 0, 1, 1};

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

    data[i] = dataword[i];

for (i = DATA_BITS; i < DATA_BITS + DIV_BITS - 1; i++)

    data[i] = 0;

for (i = 0; i < DATA_BITS + DIV_BITS - 1; i++)

    temp[i] = data[i];

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

    divisor[i] = divisor_input[i];

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

    if (temp[i] == 1) {

        for (j = 0; j < DIV_BITS; j++)

            temp[i + j] ^= divisor[j];

    }

}

printf("CRC bits: ");

for (i = DATA_BITS; i < DATA_BITS + DIV_BITS - 1; i++)

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

printf("\nCodeword: ");

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

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

for (i = DATA_BITS; i < DATA_BITS + DIV_BITS - 1; i++)

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

printf("\n");

return 0;

}

```

(b)

```
main.cpp
1  #include <stdio.h>
2
3  void printBinary(unsigned int num, int bits) {
4      for (int i = bits - 1; i >= 0; i--) {
5          printf("%d", (num >> i) & 1);
6      }
7  }
8  int main() {
9      int i;
10     unsigned int sum = 0, checksum;
11     const int m = 8;
12     const int n = 5;
13     unsigned int data[n];
14     printf("Enter %d data values (each 0-255):\n", n);
15     for (i = 0; i < n; i++) {
16         scanf("%u", &data[i]);
17         sum += data[i];
18     }
19     unsigned int mask = (1 << m) - 1;
20     while (sum >> m) {
21         sum = (sum & mask) + (sum >> m);
22     }
23     checksum = ~sum & mask;
24     printf("Checksum (8-bit) = ");
25     printf("%u\n", checksum);
26     printf("Checksum (8-bit) = ");
27     printBinary(checksum, m);
28     printf("\n");
29     return 0;
30 }
```

```
#include <stdio.h>
```

```
void printBinary(unsigned int num, int bits) {
```

```
    for (int i = bits - 1; i >= 0; i--) {
```

```
        printf("%d", (num >> i) & 1);
```

```
    }
```

```
}
```

```
int main() {
```

```
    int i;
```

```
    unsigned int sum = 0, checksum;
```



```

const int m = 8;

const int n = 5;

unsigned int data[n];

printf("Enter %d data values (each 0-255):\n", n);

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

    scanf("%u", &data[i]);

    sum += data[i];

}

unsigned int mask = (1 << m) - 1;

while (sum >> m) {

    sum = (sum & mask) + (sum >> m);

}

checksum = ~sum & mask;

printf("Checksum (8-bit) = ");

printf("%u\n", checksum);

printf("Checksum (8-bit) = ");

printBinary(checksum, m);

printf("\n");

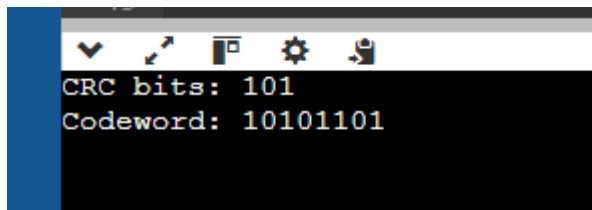
return 0;

}

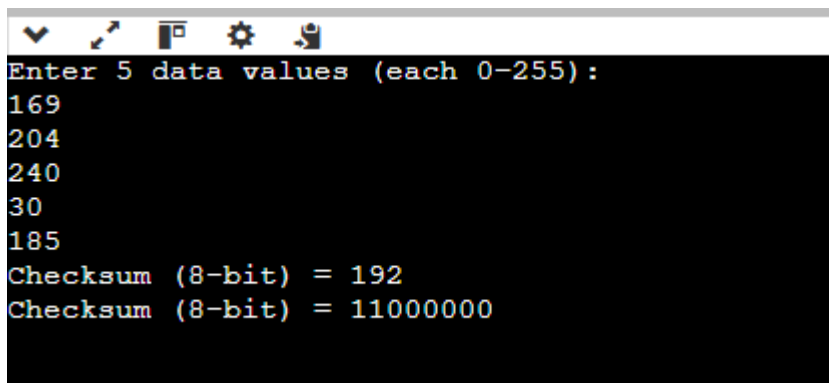
```

Output:

(a)



(b)



Result:

The output of the code is verified and the following written code runs on the system c/cpp compiler.