
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# School of Information Technology and Engineering

## Lab Assessment-II, AUGUST 2020

## B.Tech., Fall-2020-2021

| | |
|---|---|
| NAME | PRIYAL BHARDWAJ |
| REG. NO. | 18BIT0272 |
| COURSE CODE | ITE3001 |
| COURSE NAME | DATA COMMUNICATION & COMPUTER NETWORKS |
| SLOT | L15+L16 |
| FACULTY | Prof. DINAKARAN MURUGANANDAM |

**1. Assuming that your error detection method is following even parity, find the parity bit for each of the following data units.**

```
#PARITY CHECK FUNCTION
def chk_parity(bins):
    sum = 0
    for i in bins:
        if(i=='1'):
            sum+=1
    return sum%2

server_bins = input("Enter the dataword: ")
if(chk_parity(server_bins)==0):
     parity_bit = 0
     server_bins += '0'
else:
     parity_bit = 1
     server_bins += '1'

print ("The Parity bit will be:", parity_bit)
print ("The resultant codeword will be:", server_bins)
```

**a. 1001011**

→ Parity bit: 0

```
PRIYAL BHARDWAJ@LAPTOP-4OH6KB2G MINGW64 /c/CrashCourse
$ "C:/Users/PRIYAL BHARDWAJ/AppData/Local/Programs/Python/Python38/python.exe"
c:/CrashCourse/dccn1.py
Enter the dataword: 1001011
The Parity bit will be: 0
The resultant codeword will be: 10010110
```

**b. 0001100**

→ Parity bit: 0

```
PRIYAL BHARDWAJ@LAPTOP-4OH6KB2G MINGW64 /c/CrashCourse
$ "C:/Users/PRIYAL BHARDWAJ/AppData/Local/Programs/Python/Python38/python.exe"
c:/CrashCourse/dccn1.py
Enter the dataword: 0001100
The Parity bit will be: 0
The resultant codeword will be: 00011000
```

**c. 1000000**

→ Parity bit: 1

```
PRIYAL BHARDWAJ@LAPTOP-4OH6KB2G MINGW64 /c/CrashCourse
$ "C:/Users/PRIYAL BHARDWAJ/AppData/Local/Programs/Python/Python38/python.exe"
c:/CrashCourse/dccn1.py
Enter the dataword: 1000000
The Parity bit will be: 1
The resultant codeword will be: 10000001
```

**d. 1110111**

→ Parity bit: 0

```
PRIYAL BHARDWAJ@LAPTOP-4OH6KB2G MINGW64 /c/CrashCourse
$ "C:/Users/PRIYAL BHARDWAJ/AppData/Local/Programs/Python/Python38/python.exe"
c:/CrashCourse/dccn1.py
Enter the dataword: 1110111
The Parity bit will be: 0
The resultant codeword will be: 11101110
```

**2. A sender needs to send the four data items Ox3456, OxABCC, Ox02BC, and OxEEEE.**

```
def flip(c):
    return '1' if (c == '0') else '0'

def chksum(bins):
    n=len(bins)
    bin1s=''
    for i in range(n):
        bin1s += flip(bins[i])
    return bin1s

#Hexanumeric addition
def hexchk():
    l=5
    sum = 0
    max = 0
    for i in range(n):
        x=int(input(),16)
        sum+=x
        if max<len(format(x,'b')):
            max = len(format(x,'b'))
    sum = format(sum,'b')
    if max < len(sum):
        sum = format(int(sum[:len(sum)-max],16) + int(sum[-max:],16), 'x')
    return sum

print("Sender's Side\n")
n = int(input('Enter total no. of inputs: '))
print("\nEnter the datawords:")
sendersum = chksum(hexchk())
print("Checksum on Sender Site: " + sendersum + "\n\n")
print("Receiver's Side\n\nEnter the datawords:")
receiversum = chksum(hexchk())
print("Checksum on Receiver Site: " + receiversum)
print('No Error') if(sendersum==receiversum) else print('Error')
```

**a. Find the checksum at the sender site.**

→ Checksum at Sender Site: 0010111000110010

```
PRIYAL BHARDWAJ@LAPTOP-4OH6KB2G MINGW64 /c/CrashCourse
$ "C:/Users/PRIYAL BHARDWAJ/AppData/Local/Programs/Python/Python38/python.exe"
c:/CrashCourse/dccn1.py
Sender's Side

Enter total no. of inputs: 4

Enter the datawords:
3456
ABCC
02BC
EEEE
Checksum on Sender Site: 0010111000110010
```

**b. Find the checksum at the receiver site if there is no error.**

→ Checksum at Receiver Site: 0010111000110010

```
Receiver's Side

Enter the datawords:
3456
ABCC
02BC
EEEE
Checksum on Receiver Site: 0010111000110010
No Error
```

**c. Find the checksum at the receiver site if the second data item is changed to OxABCE.**

→ Checksum at Sender Site: 0010111000110010

```
Receiver's Side

Enter the datawords:
3456
ABCE
02BC
EEEE
Checksum on Receiver Site: 0010111000110000
Error
```

**PTO**

**d. Find the checksum at the receiver site if the second data item is changed to OxABCE and the third data item is changed to Ox02BA.**

→ Checksum at Sender Site: 0010111000110010

```
Receiver's Side

Enter the datawords:
3456
ABCE
02BA
EEEE
Checksum on Receiver Site: 0010111000110010
No Error
```

**3. Given the dataword 1010011110 and the divisor 10111,**

```
#mod 2 addition or XOR
def mod2(x,y):
    mod2bit=""
    for i in range(1, len(y)):
        if(x[i]==y[i]):
            mod2bit=mod2bit+'0'
        else:
            mod2bit=mod2bit+'1'
    return mod2bit

#performing crc division
def crc(dividend,divisor):
    l=5
    rem=dividend[0:l]
    while(l<len(dividend)):
        if rem[0] == '1':
            rem=mod2(divisor,rem)+dividend[l]
        else:
            rem = mod2('0'*5, rem) + dividend[l]
        l+= 1
    if(rem[0]=='1'):
        rem=mod2(divisor, rem)
    else:
        rem=mod2('0'*5, rem)
    return rem      #returning remainder

data = input("Enter the Dataword(Binary): ")
divisor = input("Enter the Divisor(5 bits): ")
ud = data+'0000'    #update the data by appending 0000
print('The updated dataword: ',ud)
data = data+crc(ud, divisor)
print('The codeword after Binary division: ', data)
```

```
#Error detection
inp = input('Error or not(Y/N): ')
if(inp.lower() == 'y'): #error
    data = input("Enter the Dataword(Binary): ")
    print("Remainder: ", crc(data, divisor))
    if(crc(data, divisor)=='0000'):
        print("Status: No Error")
    else:
        print("Status: Error")
else:               #no error
    print('Remainder:', crc(data, divisor))
    print('Status: No Error')
```

**a. Generate the codeword at the sender site (using binary division).**

→ Codeword: 10100111101010

**b. Check the codeword at the receiver site (assume no error).**

→ After assuming no error, we get remainder 0000 on checking the codeword.

```
PRIYAL BHARDWAJ@LAPTOP-4OH6KB2G MINGW64 /c/CrashCourse
$ "C:/Users/PRIYAL BHARDWAJ/AppData/Local/Programs/Python/Python38/python.exe"
Enter the Dataword(Binary): 1010011110
Enter the Divisor(5 bits): 10111
The updated dataword:  10100111100000
The codeword after Binary division:  10100111101010
Error or not(Y/N): N
Remainder: 0000
Status: No Error
```

**********