

# Assignment 4

Anjishnu Mukherjee B05-511017020 (510517086)

## 4a.

The aim of this programming assignment is to **simulate a rudimentary time division multiplexing mechanism**. Multiplexing STS-1 multiple data streams, called tributaries, plays an important role in SONET. A 3:1 multiplexer multiplex three input STS-1 tributaries onto one output STS-3 stream. This multiplexing is done byte for byte. That is, the first three output bytes are the first bytes of tributaries 1, 2, and 3, respectively. The next three output bytes are the second bytes of tributaries 1, 2, and 3, respectively, and so on. Write a program that simulates this 3:1 multiplexer. Your program should consist of five processes. The main process creates four processes, one each for the three STS-1 tributaries and one for the multiplexer. Each tributary process reads in an STS-1 frame from an input file as a sequence of 100 bytes. They send their frames (byte by byte) to the multiplexer process. The multiplexer process receives these bytes and outputs an STS-3 frame (byte by byte) by writing it to standard output. Use pipes for communication STS and the multiplexer processes.

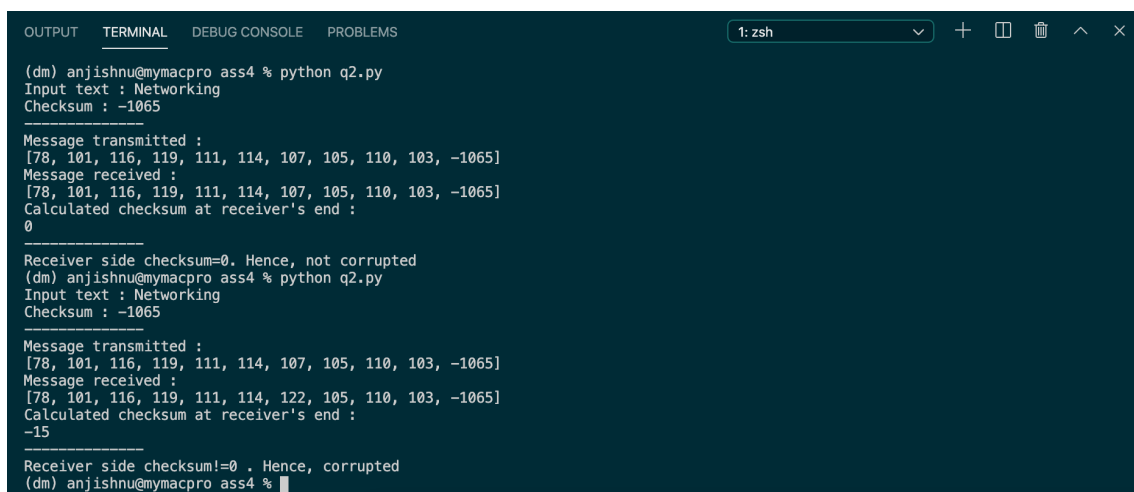
I have implemented the solution in Python using only in-built libraries for system calls and random number generation purposes. A sample output for a reduced number of input bytes to each source is given below.

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS  1: zsh
(dm) anjishnu@mymacpro ass4 % python q1.py
Input text : FN3N0SM8JL
Input text : 6CE5J3R64P
Input text : AM85BC7R9V
Frame 1 : F6A
Frame 2 : NCM
Frame 3 : 3E8
Frame 4 : N55
Frame 5 : 0JB
Frame 6 : S3C
Frame 7 : MR7
Frame 8 : 86R
Frame 9 : J49
Frame 10 : LPV
(dm) anjishnu@mymacpro ass4 %
```

#### 4b.

The objective of this programming assignment is to **simulate the usage of checksum as an error detection technique in data transmission**. Suppose the payload portion of a packet contains 10 bytes consisting of the 8-bit unsigned binary ASCII representation of string "Networking." Write a program to compute the Internet checksum for this data. Your program should have three functions. The first function reads the data from an input file and computes the checksum. The second function simulates a coin toss phenomenon to determine if the packet will be corrupted during transmission. If it is corrupted, change one or more bytes. The third function again computes the checksum and decides if the data is to be accepted or discarded.

I have implemented the solution in Python using only in-built libraries for system calls and random number generation purposes. A sample output for a reduced number of input bytes to each source is given below.



```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
1: zsh

(dm) anjishnu@mymacpro ass4 % python q2.py
Input text : Networking
Checksum : -1065

-----
Message transmitted :
[78, 101, 116, 119, 111, 114, 107, 105, 110, 103, -1065]
Message received :
[78, 101, 116, 119, 111, 114, 107, 105, 110, 103, -1065]
Calculated checksum at receiver's end :
0

-----
Receiver side checksum=0. Hence, not corrupted
(dm) anjishnu@mymacpro ass4 % python q2.py
Input text : Networking
Checksum : -1065

-----
Message transmitted :
[78, 101, 116, 119, 111, 114, 107, 105, 110, 103, -1065]
Message received :
[78, 101, 116, 119, 111, 114, 122, 105, 110, 103, -1065]
Calculated checksum at receiver's end :
-15

-----
Receiver side checksum!=0 . Hence, corrupted
(dm) anjishnu@mymacpro ass4 %
```