

## CN Computer Networks Assignment 2

This is my report for the second assignment of the Computer Networks course. The report is written in markdown and can be viewed in any markdown viewer. The assignment expected us to make a TCP receiver.

### Part 1

Part 1 - Required to us to implement a `ByteStream` class which would be used by the TCP receiver to store the incoming data. The class was implemented using a deque. The class has the following methods:

1. `ByteStream()` - Constructor for the class
2. `write(string)` - Pushes a string to the back of the deque
3. `peak_output(int)` - Returns the string stored in the deque from the front till the given index
4. `read(int)` - Returns the string stored in the deque from the front till the given index and removes it from the deque
5. `pop_output(int)` - Removes the string stored in the deque from the front till the given index
6. `buffer_empty()` - Returns true if the deque is empty
7. `size()` - Returns the size of the deque
8. `eof()` - Returns true if stream has ended
9. `bytes_written()` - Returns the number of bytes written to the stream
10. `bytes_read()` - Returns the number of bytes read from the stream

The image below shows the output of the test cases for the `ByteStream` class.

```
Test project /home/dev/CN/CSE232-ComputerNetworks-Assignments/assignment2/build
Start 5: byte_stream_construction
1/5 Test #5: byte_stream_construction ..... Passed    0.00 sec
Start 6: byte_stream_one_write
2/5 Test #6: byte_stream_one_write ..... Passed    0.00 sec
Start 7: byte_stream_two_writes
3/5 Test #7: byte_stream_two_writes ..... Passed    0.00 sec
Start 8: byte_stream_capacity
4/5 Test #8: byte_stream_capacity ..... Passed    0.70 sec
Start 9: byte_stream_many_writes
5/5 Test #9: byte_stream_many_writes ..... Passed    0.00 sec

100% tests passed, 0 tests failed out of 5

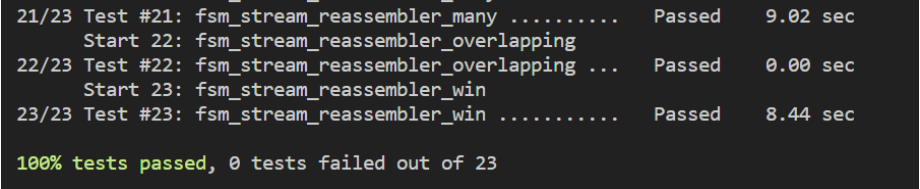
Total Test time (real) =  0.72 sec
```

Figure 1: test\_case

## Part 2 and 3

Part 2 - Required us to implement a Reassembler class which would be used by the TCP receiver to reassemble the incoming data. Then we had to implement a TCP receiver which would use the ByteStream and Reassembler class to reassemble the incoming data.

The image below shows the output of the test cases for the Reassembler class.



```
21/23 Test #21: fsm_stream_reassembler_many ..... Passed    9.02 sec
      Start 22: fsm_stream_reassembler_overlapping
22/23 Test #22: fsm_stream_reassembler_overlapping ... Passed    0.00 sec
      Start 23: fsm_stream_reassembler_win
23/23 Test #23: fsm_stream_reassembler_win ..... Passed    8.44 sec

100% tests passed, 0 tests failed out of 23
```