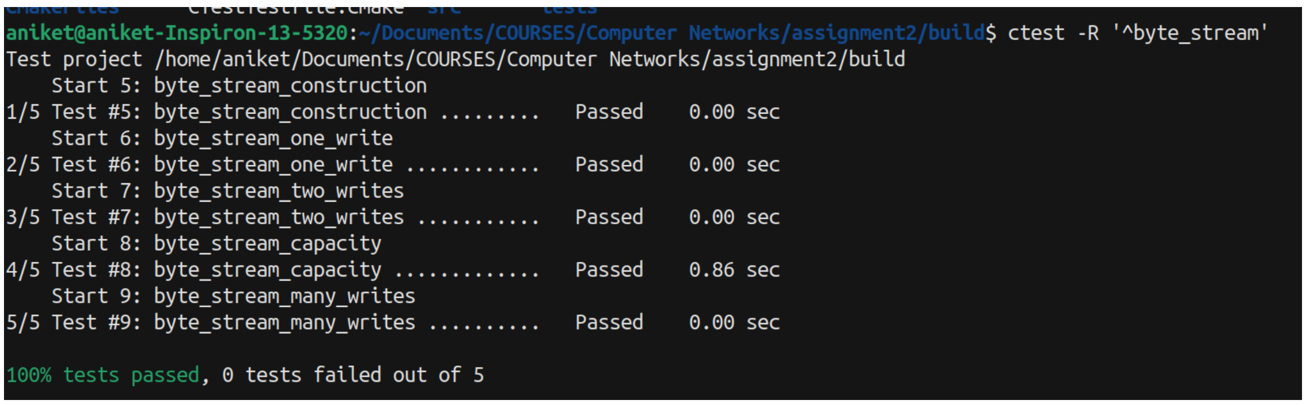
CN Assignment 2

PART A: Byte stream implementation
REPORT

Successfull run of given test cases of byte_stream.cc implementation.



```
aniket@aniket-Inspiron-13-5320:~/Documents/COURSES/Computer Networks/assignment2/build$ ctest -R '^byte_stream'
Test project /home/aniket/Documents/COURSES/Computer Networks/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    0.86 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
```

1. Bytes are written on the input side and read out from the output side (use a data structure that allows pushing the byte from one side and popping from the other side).
Solution: Used Deque data structure which allows input and output from both ends
I used it as to input from one end, and other end as output

2. The byte stream is finite. The writer can end the input and no more bytes can be written.
I implemented, the deque having a fixed capacity, which is given to it, when creating the object
3. When the reader has read to the end of the stream, it will reach EOF (End of File), that is no more available bytes to read

Solution: And a feature where user can end or decide if it does not want to input any other bytes,
Or it reaches the end of capacity, thus ignoring all bytes after reaching the buffer capacity

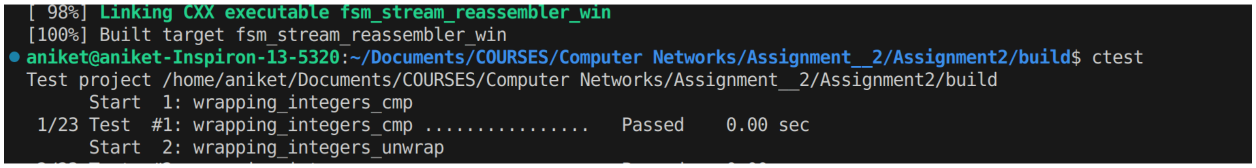
4. Your abstraction will also be initialized with a particular capacity which limits the total amount of bytes that can be held in memory at once (which are not read yet).
Solution: When the object is created, user initialised the capacity of the buffer as well.
That if it gets fill, the EOF is achieved, and no more bytes can be written.

5. The writer would not be allowed to write into the byte stream if it exceeds the storage Capacity.
Solution: I put a if check, in the beginning of write function, If capacity is full it can add any more bits.

6. As the reader reads bytes from the stream, the writer is allowed to write more.
Solution: To achieve this, i am removing the bytes that has been read, so that the buffer get more space, and we can add more bytes into it.

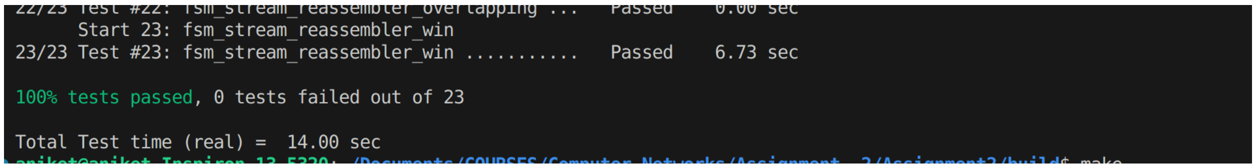
Computer Networks
Assignment 2
Building a TCP receiver
Deadline 2(Part B- Reassembler and Part C -TCP receiver)

☐ Checking all test cases using: ctest command



```
[ 98%] Linking CXX executable fsm_stream_reassembler_win
[100%] Built target fsm_stream_reassembler_win
aniket@aniket-Inspiron-13-5320:~/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build$ ctest
Test project /home/aniket/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build
      Start  1: wrapping_integers_cmp
 1/23 Test  #1: wrapping_integers_cmp ................   Passed    0.00 sec
      Start  2: wrapping_integers_unwrap
```

ctest final status:

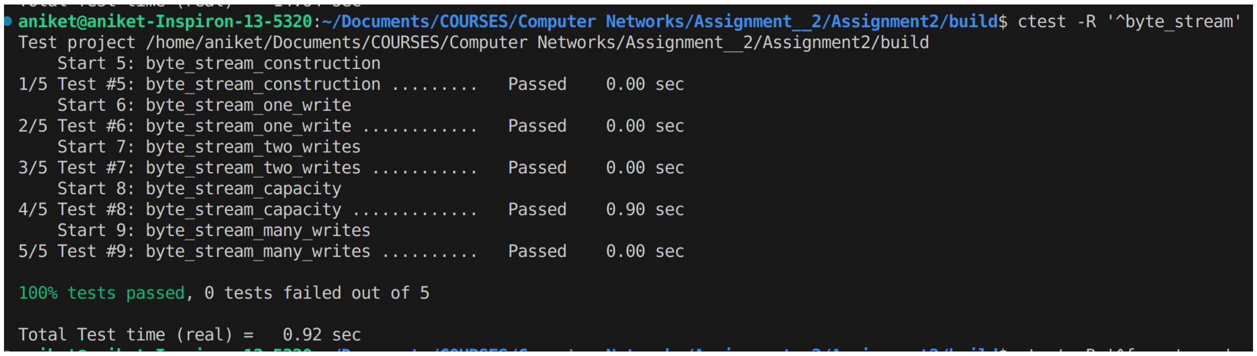


```
22/23 Test #22: fsm_stream_reassembler_overlapping ...   Passed    0.00 sec
      Start 23: fsm_stream_reassembler_win
23/23 Test #23: fsm_stream_reassembler_win ...........   Passed    6.73 sec

100% tests passed, 0 tests failed out of 23

Total Test time (real) =  14.00 sec
```

☐ Checking individual Byte_stream test cases using: ctest -R ‘^byte_stream’

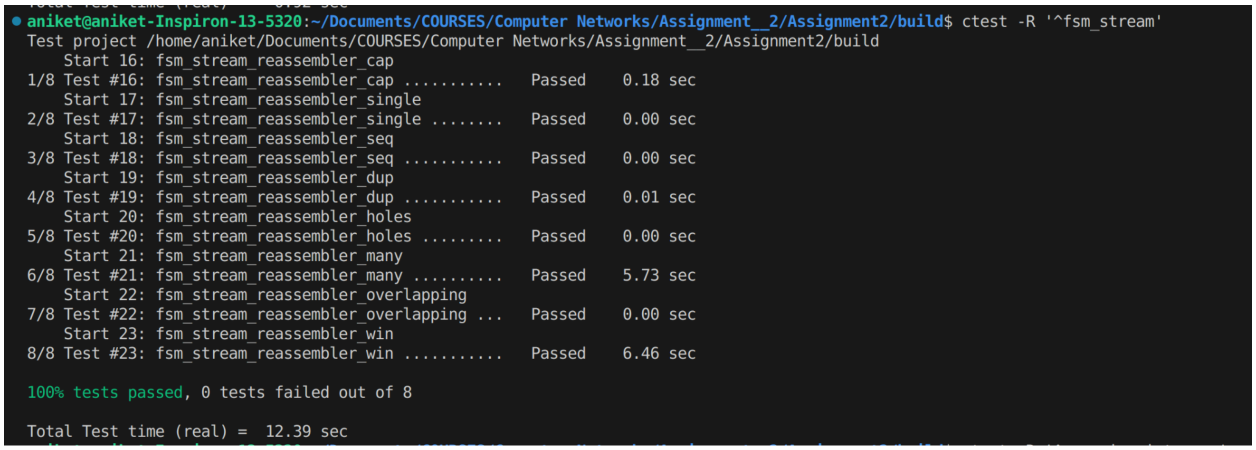


```
aniket@aniket-Inspiron-13-5320:~/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build$ ctest -R '^byte_stream'
Test project /home/aniket/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build
    Start 5: byte_stream_construction
1/5 Test #5: byte_stream_construction .........   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    0.90 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) =   0.92 sec
```

☐ Checking individual stream_reassembler test cases using: ctest -R ‘^fsm_stream’

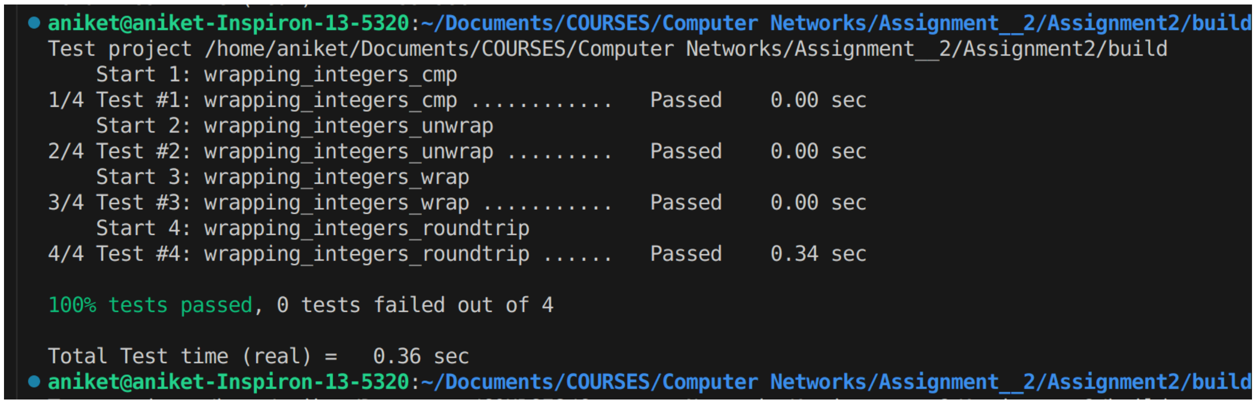


```
aniket@aniket-Inspiron-13-5320:~/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build$ ctest -R '^fsm_stream'
Test project /home/aniket/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build
    Start 16: fsm_stream_reassembler_cap
1/8 Test #16: fsm_stream_reassembler_cap ...........   Passed    0.18 sec
    Start 17: fsm_stream_reassembler_single
2/8 Test #17: fsm_stream_reassembler_single ........   Passed    0.00 sec
    Start 18: fsm_stream_reassembler_seq
3/8 Test #18: fsm_stream_reassembler_seq ...........   Passed    0.00 sec
    Start 19: fsm_stream_reassembler_dup
4/8 Test #19: fsm_stream_reassembler_dup ...........   Passed    0.01 sec
    Start 20: fsm_stream_reassembler_holes
5/8 Test #20: fsm_stream_reassembler_holes .........   Passed    0.00 sec
    Start 21: fsm_stream_reassembler_many
6/8 Test #21: fsm_stream_reassembler_many ..........   Passed    5.73 sec
    Start 22: fsm_stream_reassembler_overlapping
7/8 Test #22: fsm_stream_reassembler_overlapping ...   Passed    0.00 sec
    Start 23: fsm_stream_reassembler_win
8/8 Test #23: fsm_stream_reassembler_win ...........   Passed    6.46 sec

100% tests passed, 0 tests failed out of 8

Total Test time (real) =  12.39 sec
```

☐ Checking individual wrapping_integers test cases using ctest -R ‘^wrapping_integers’

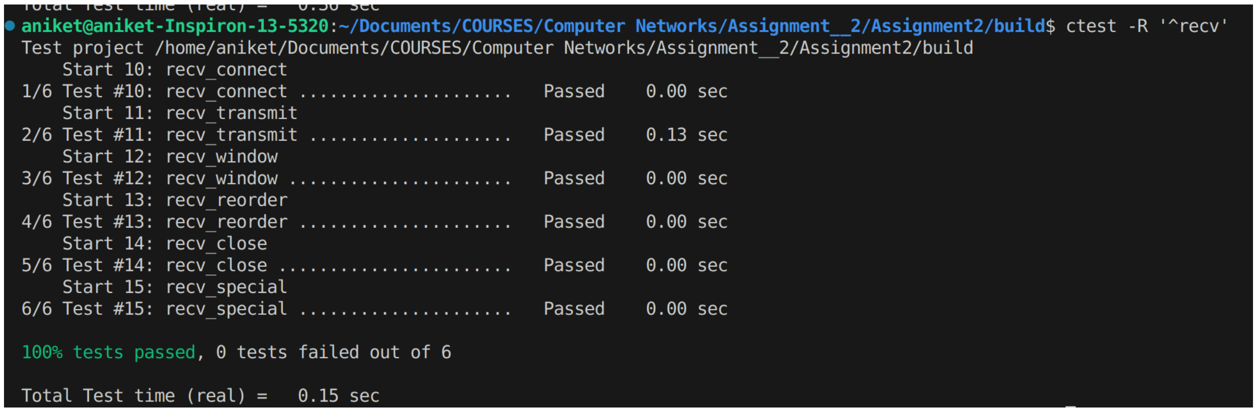


```
● aniket@aniket-Inspiron-13-5320:~/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build
  Test project /home/aniket/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build
      Start 1: wrapping_integers_cmp
  1/4 Test #1: wrapping_integers_cmp ............   Passed    0.00 sec
      Start 2: wrapping_integers_unwrap
  2/4 Test #2: wrapping_integers_unwrap .........   Passed    0.00 sec
      Start 3: wrapping_integers_wrap
  3/4 Test #3: wrapping_integers_wrap ...........   Passed    0.00 sec
      Start 4: wrapping_integers_roundtrip
  4/4 Test #4: wrapping_integers_roundtrip ......   Passed    0.34 sec

  100% tests passed, 0 tests failed out of 4

  Total Test time (real) =   0.36 sec
● aniket@aniket-Inspiron-13-5320:~/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build
```

☐ Checking individual TCP receiver test cases using: ctest -R ‘^recv’



```
● aniket@aniket-Inspiron-13-5320:~/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build$ ctest -R '^recv'
  Test project /home/aniket/Documents/COURSES/Computer Networks/Assignment__2/Assignment2/build
      Start 10: recv_connect
  1/6 Test #10: recv_connect .....................   Passed    0.00 sec
      Start 11: recv_transmit
  2/6 Test #11: recv_transmit ....................   Passed    0.13 sec
      Start 12: recv_window
  3/6 Test #12: recv_window ......................   Passed    0.00 sec
      Start 13: recv_reorder
  4/6 Test #13: recv_reorder .....................   Passed    0.00 sec
      Start 14: recv_close
  5/6 Test #14: recv_close .......................   Passed    0.00 sec
      Start 15: recv_special
  6/6 Test #15: recv_special .....................   Passed    0.00 sec

  100% tests passed, 0 tests failed out of 6

  Total Test time (real) =   0.15 sec
```

All functions have been wholly explained, and the reason and theory behind it have been written in a well-structured manner in the code files themselves.