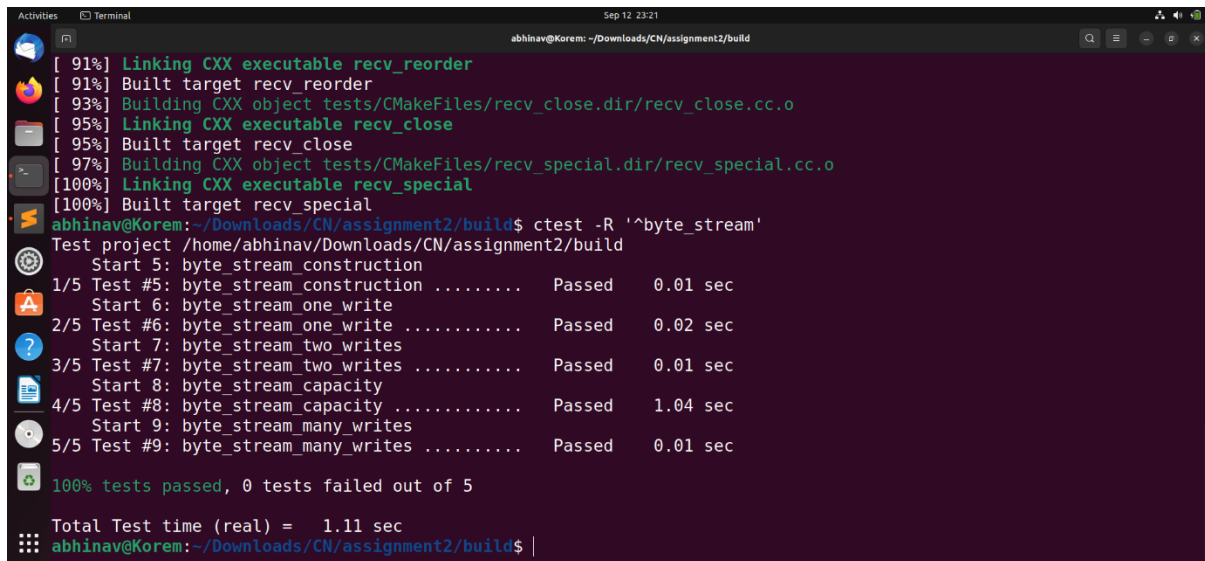


Assignment – 2

Screenshot of all the test case of Bytestream passing:

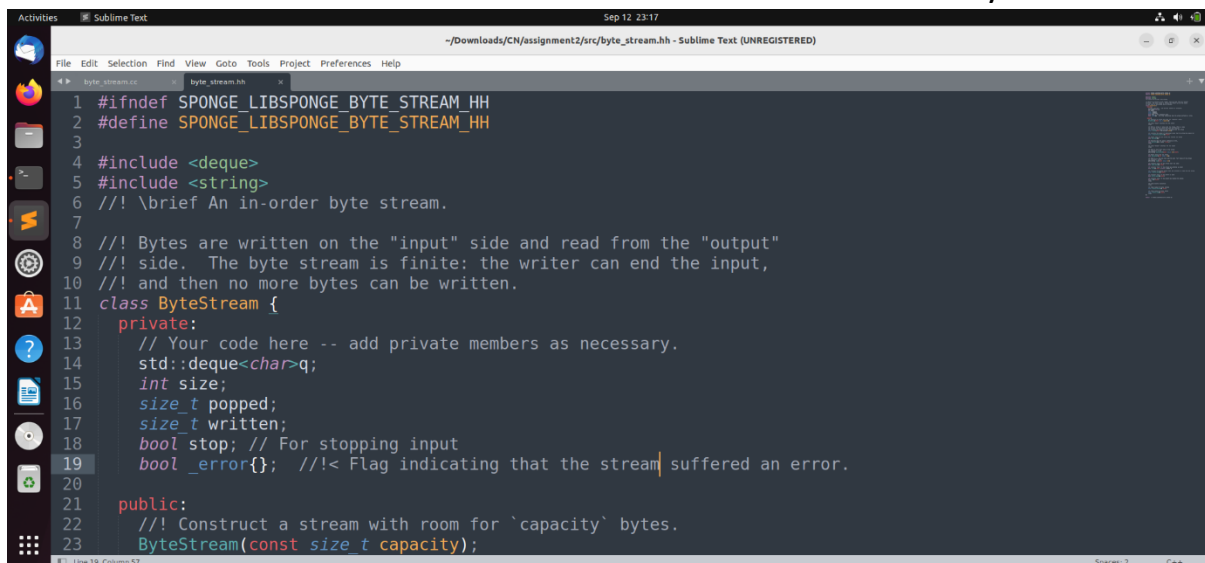


```
abhinav@Korem: ~/Downloads/CN/assignment2/build
[ 91%] Linking CXX executable recv_reorder
[ 91%] Built target recv_reorder
[ 93%] Building CXX object tests/CMakeFiles/recv_close.dir/recv_close.cc.o
[ 95%] Linking CXX executable recv_close
[ 95%] Built target recv_close
[ 97%] Building CXX object tests/CMakeFiles/recv_special.dir/recv_special.cc.o
[100%] Linking CXX executable recv_special
[100%] Built target recv_special
abhinav@Korem:~/Downloads/CN/assignment2/build$ ctest -R '^byte_stream'
Test project /home/abhinav/Downloads/CN/assignment2/build
Start 5: byte_stream_construction
1/5 Test #5: byte_stream_construction ..... Passed    0.01 sec
Start 6: byte_stream_one_write
2/5 Test #6: byte_stream_one_write ..... Passed    0.02 sec
Start 7: byte_stream_two_writes
3/5 Test #7: byte_stream_two_writes ..... Passed    0.01 sec
Start 8: byte_stream_capacity
4/5 Test #8: byte_stream_capacity ..... Passed    1.04 sec
Start 9: byte_stream_many_writes
5/5 Test #9: byte_stream_many_writes ..... Passed    0.01 sec

100% tests passed, 0 tests failed out of 5

Total Test time (real) = 1.11 sec
abhinav@Korem:~/Downloads/CN/assignment2/build$
```

These are the variables that have been used to create the class ByteStream.



```
1 #ifndef SPONGE_LIBSPONGE_BYTE_STREAM_HH
2 #define SPONGE_LIBSPONGE_BYTE_STREAM_HH
3
4 #include <deque>
5 #include <string>
6 /// \brief An in-order byte stream.
7
8 /// Bytes are written on the "input" side and read from the "output"
9 /// side. The byte stream is finite: the writer can end the input,
10 /// and then no more bytes can be written.
11 class ByteStream {
12 private:
13     // Your code here -- add private members as necessary.
14     std::deque<char> q;
15     int size;
16     size_t popped;
17     size_t written;
18     bool stop; // For stopping input
19     bool _error{}; //!< Flag indicating that the stream suffered an error.
20
21 public:
22     /// Construct a stream with room for `capacity` bytes.
23     ByteStream(const size_t capacity);
```

We have assumed that the functions other than the write, read, peek and pop are not affected by the `_error` flag being true and will function normally. The mentioned functions, however, will not work after the `_error` flag has been set to true.