My name: 李明扬

My Student number : 502024330026

This lab took me about 3 hours to do. I did attend the lab session.

1. Program Structure and Design:

- 本次实验主要要求实现`Reassembler`类，将收集到的字符串拼接在一起。需要考虑到ByteStream的容量，将超出容量部分的字符串舍弃。还需要注意是否输入了eof，在合适的时候调用`close()`。
- 我采用了STL中的`deque`类来作为本次实验的主要数据结构。我定义了两个`deque`类，分别用来存储正在重组的字符以及用`bool`来表示某个位置是否存在字符。每一个`deque`预留足够的空间来避免从中间插入字符从而提升效率。在插入字符时，将特定位置的值改为`true`，再将字符写入。在写入字节流时，只需要把`deque`中前面的字符输出即可。
- 为了使字符串按照顺序排列，我维护了一个索引变量，用以表示队列中第一个字符在整个字节流中的位置。在每次写入字节流时，对这个值进行更新。
- 为了实现`eof`的写入，我维护了另一个索引变量，用于表示`eof`在队列中的位置。当收到`eof`并将其加入缓存中时，就开始维护这个变量，每次写入字节流就更新这个变量。直到这个变量变为0，则调用`close()`表示结束。

2. Implementation Challenges:

- 本次实验最大的挑战在于如何维护许多可能会重叠的子字符串。如果将子字符串全部保存下来，会存在大量重叠，不方便写入，也不利于计算索引。于是我在插入字符串时，将字符串分解为字符来一个一个写入，从而降低了维护难度。

- 另一个挑战是对于索引的计算。每个字符在整个字节流中有一个索引，在队列中也有一个索引，需要通过计算将其一一对应。这一过程比较繁琐，需要仔细分析才能写对代码。

- 如何确定什么时候调用`close()`也是一个问题，需要确定`eof`是否被加入到了队列中，以及其在队列中的哪个位置，这个位置需要随着队列的流动而更新。

3. Remaining Bugs:

- 暂时尚未找到bug。

测试结果如下图

```
(base) li@li-System-Product-Name:~/projects/minnow$ cmake --build build --target check1
Test project /home/li/projects/minnow/build
      Start  1: compile with bug-checkers
 1/17 Test  #1: compile with bug-checkers ........   Passed    0.95 sec
      Start  3: byte_stream_basics
 2/17 Test  #3: byte_stream_basics ...............   Passed    0.01 sec
      Start  4: byte_stream_capacity
 3/17 Test  #4: byte_stream_capacity .............   Passed    0.01 sec
      Start  5: byte_stream_one_write
 4/17 Test  #5: byte_stream_one_write ............   Passed    0.01 sec
      Start  6: byte_stream_two_writes
 5/17 Test  #6: byte_stream_two_writes ...........   Passed    0.01 sec
      Start  7: byte_stream_many_writes
 6/17 Test  #7: byte_stream_many_writes ..........   Passed    0.03 sec
      Start  8: byte_stream_stress_test
 7/17 Test  #8: byte_stream_stress_test ..........   Passed    0.01 sec
      Start  9: reassembler_single
 8/17 Test  #9: reassembler_single ...............   Passed    0.04 sec
      Start 10: reassembler_cap
 9/17 Test #10: reassembler_cap ..................   Passed    0.01 sec
      Start 11: reassembler_seq
10/17 Test #11: reassembler_seq ..................   Passed    0.04 sec
      Start 12: reassembler_dup
11/17 Test #12: reassembler_dup ..................   Passed    0.05 sec
      Start 13: reassembler_holes
12/17 Test #13: reassembler_holes ................   Passed    0.05 sec
      Start 14: reassembler_overlapping
13/17 Test #14: reassembler_overlapping ..........   Passed    0.01 sec
      Start 15: reassembler_win
14/17 Test #15: reassembler_win ..................   Passed    2.51 sec
      Start 37: compile with optimization
15/17 Test #37: compile with optimization ........   Passed    0.58 sec
      Start 38: byte_stream_speed_test
            ByteStream throughput: 6.08 Gbit/s
16/17 Test #38: byte_stream_speed_test ...........   Passed    0.07 sec
      Start 39: reassembler_speed_test
            Reassembler throughput: 0.99 Gbit/s
17/17 Test #39: reassembler_speed_test ...........   Passed    0.22 sec

100% tests passed, 0 tests failed out of 17

Total Test time (real) =   4.61 sec
Built target check1
```