

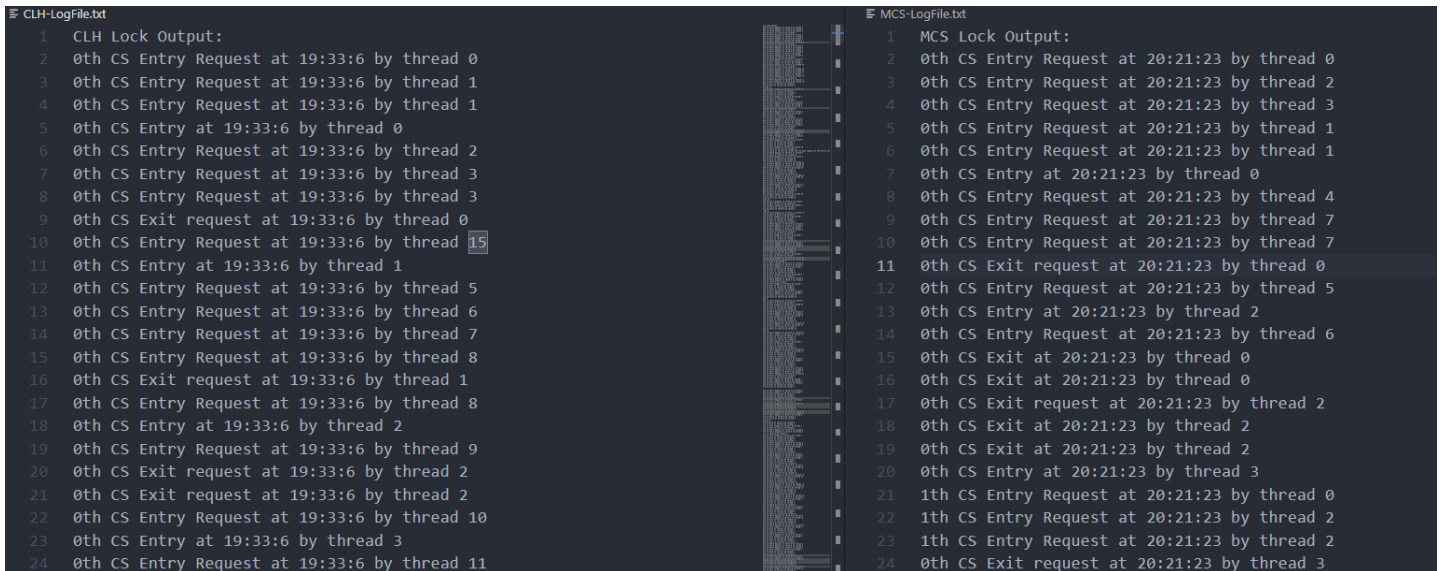
# Parallel & Concurrent Programming

## Programming Assignment 5 Report

Thupten Dukpa  
ES20BTECH11029

In each of the CLHLock and MCSLock programs, there is a Lock class with pure virtual functions lock() and unlock() which are inherited and overwritten. QNode is defined as a struct. The CLHLock and MCSLock classes are then implemented according to the book. The main function reads the inputs, creates threads each of which calls the testCS() function.

### Sample output



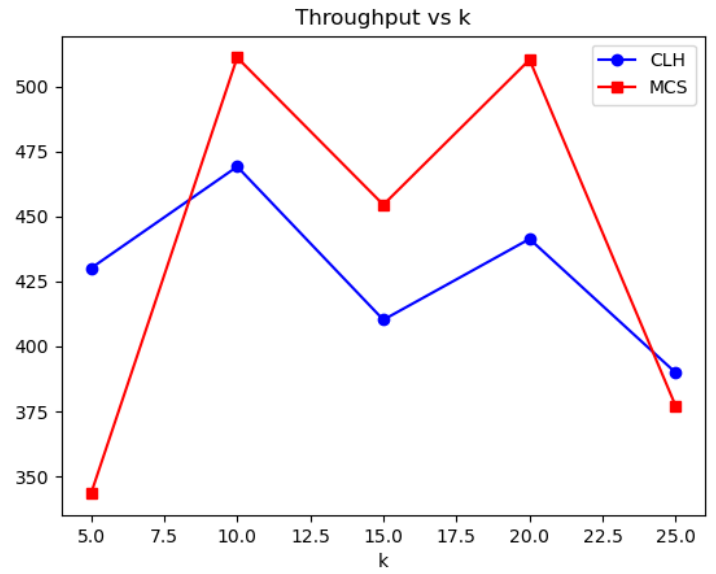
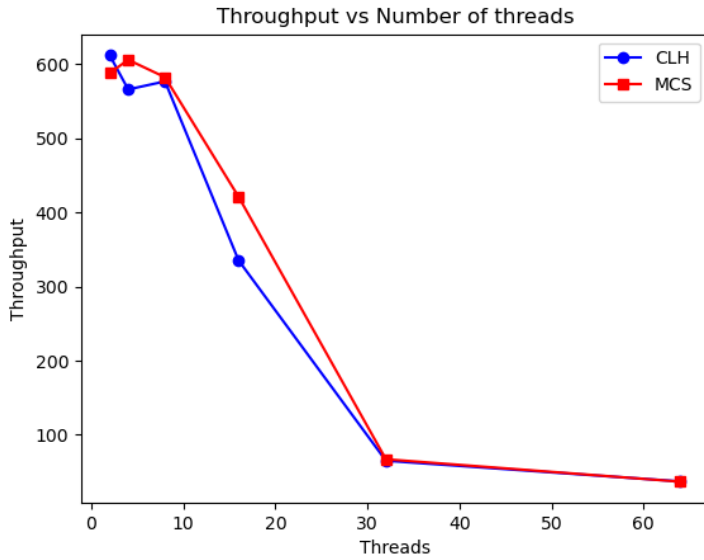
The image shows two side-by-side terminal windows. The left window, titled 'CLH-LogFile.txt', displays the output of a CLH lock test. It shows a sequence of '0th CS Entry Request' and '0th CS Exit request' messages for threads 0 through 11, all occurring at the same timestamp '19:33:6'. The right window, titled 'MCS-LogFile.txt', displays the output of an MCS lock test. It shows a sequence of '0th CS Entry Request' and '0th CS Exit request' messages for threads 0 through 3, all occurring at the same timestamp '20:21:23'. Both outputs demonstrate that only one thread enters the critical section at a time, as evidenced by the sequential nature of the entry and exit requests.

```
CLH-LogFile.txt
1 CLH Lock Output:
2 0th CS Entry Request at 19:33:6 by thread 0
3 0th CS Entry Request at 19:33:6 by thread 1
4 0th CS Entry Request at 19:33:6 by thread 1
5 0th CS Entry at 19:33:6 by thread 0
6 0th CS Entry Request at 19:33:6 by thread 2
7 0th CS Entry Request at 19:33:6 by thread 3
8 0th CS Entry Request at 19:33:6 by thread 3
9 0th CS Exit request at 19:33:6 by thread 0
10 0th CS Entry Request at 19:33:6 by thread 15
11 0th CS Entry at 19:33:6 by thread 1
12 0th CS Entry Request at 19:33:6 by thread 5
13 0th CS Entry Request at 19:33:6 by thread 6
14 0th CS Entry Request at 19:33:6 by thread 7
15 0th CS Entry Request at 19:33:6 by thread 8
16 0th CS Exit request at 19:33:6 by thread 1
17 0th CS Entry Request at 19:33:6 by thread 8
18 0th CS Entry at 19:33:6 by thread 2
19 0th CS Entry Request at 19:33:6 by thread 9
20 0th CS Exit request at 19:33:6 by thread 2
21 0th CS Exit request at 19:33:6 by thread 2
22 0th CS Entry Request at 19:33:6 by thread 10
23 0th CS Entry at 19:33:6 by thread 3
24 0th CS Entry Request at 19:33:6 by thread 11

MCS-LogFile.txt
1 MCS Lock Output:
2 0th CS Entry Request at 20:21:23 by thread 0
3 0th CS Entry Request at 20:21:23 by thread 2
4 0th CS Entry Request at 20:21:23 by thread 3
5 0th CS Entry Request at 20:21:23 by thread 1
6 0th CS Entry Request at 20:21:23 by thread 1
7 0th CS Entry at 20:21:23 by thread 0
8 0th CS Entry Request at 20:21:23 by thread 4
9 0th CS Entry Request at 20:21:23 by thread 7
10 0th CS Entry Request at 20:21:23 by thread 7
11 0th CS Exit request at 20:21:23 by thread 0
12 0th CS Entry Request at 20:21:23 by thread 5
13 0th CS Entry at 20:21:23 by thread 2
14 0th CS Entry Request at 20:21:23 by thread 6
15 0th CS Exit at 20:21:23 by thread 0
16 0th CS Exit at 20:21:23 by thread 0
17 0th CS Exit request at 20:21:23 by thread 2
18 0th CS Exit at 20:21:23 by thread 2
19 0th CS Exit at 20:21:23 by thread 2
20 0th CS Entry at 20:21:23 by thread 3
21 1th CS Entry Request at 20:21:23 by thread 0
22 1th CS Entry Request at 20:21:23 by thread 2
23 1th CS Entry Request at 20:21:23 by thread 2
24 0th CS Exit request at 20:21:23 by thread 3
```

It is clear that in each of these files, no other thread enters the critical section when a particular thread holds it, i.e, there is no 'Entry' or 'Exit Request' interleaving with other threads.

## Throughput vs number of threads and k:



## Average Entry time vs number of threads and k:

