# Programming Assignment 5
# Comparison of TAS, TTAS, CLH and MCS Locks
Submission Date: 17th April 2022, 9:00 pm

**Goal:** The goal of this assignment is to implement the two locking operations discussed in the class: TAS, TTAS, CLH and MCS Locks. Then compare the performance of these locks by measuring two parameters: average and worst-case waiting time for threads to obtain the locks. You have to implement these algorithms **in C++**.

**Details.** As explained above, you have to implement two lockins algorithms: TAS, TTAS, CLH, MCS Locks. You have to implement them **in C++**. Each locking algorithm implements a class consisting of two methods: lock and unlock. Your program will read the input from the file and write the output to the file as shown in the example below.

To test the performance of locking algorithms, develop an application, lock-test is as follows (which is same as Assignment-2). Once, the program starts, it creates $n$ threads. Each of these threads, will enter critical section (CS) $k$ times. The pseudocode of the test function is as follows:

Listing 1: main thread

```
1  void main()
2  {
3      ...
4      ...
5      // Declare a lock object which is accessed from all the threads
6      Lock Test = new Lock();
7      ...
8      ...
9      create n testCS threads;
10 }
```

Listing 2: testCS thread

```
1
2  void testCS()
3  {
4      id = thread.getID();
5      for (i=0; i < k; i++)
6      {
7          reqEnterTime = getSysTime();
8          cout << i << "th CS Entry Request at " << reqEnterTime << " by thread "
9          << id;
10         Test.lock();
11         actEnterTime = getSysTime();
12         cout << i << "th CS Entery at " << actEnterTime << " by thread " << id;
13         sleep(t1);
14         reqExitTime = getSysTime();
```

```
15          cout << i << "th CS Exit Request at " << reqExitTime << " by thread "
16          << id;
17          Test.unlock();
18          actExitTime = getSysTime();
19          cout << i << "th CS Exit at " << actExitTime << " by thread " << id;
20          sleep(t2);
21      }
22  }
```

**Description of the Test Program:** The description is self-explanatory. As can be seen, each thread invokes testCS function. When a thread wishes to enter the Critical Section (CS), it stores the time as reqEnterTime (or request Enter Time). And once the thread enters the CS, it records the time as actEnterTime (actual CS Entry Time). The difference between these times actEnterTime - reqEnterTime is the time taken by the thread to enter the CS which we denote as *csEnterTime*.

Similarly, when a thread wishes to leave the CS, it records the time as reqExitTime (request Exit Time). Once it leaves the CS, it again records the time as actExitTime (request Exit Time). The difference between these times actExitTime - reqExitTime is the time taken by the thread to exit the CS which we denote as *csExitTime*.

If the lock and unlock functions work correctly then the display messages 2 and 3 of every thread will work correctly without any interleaving. Seeing these messages one can be sure of the correctness of the lock and unlock. Note that the message 1 and 4 can interleave and hence may be commented out to check the correctness.

Here $t1$ and $t2$ are delay values that are exponentially distributed with an average of $\lambda 1, \lambda 2$ seconds (and not milli-seconds). The objective of having these time delays is to simulate that these threads are performing some complicated time consuming tasks.

The $Test$ variable declared in line 6 declared in main is an instance of Lock class and is accessible by all threads.

**Input:** The input to the program will be a file, named inp-params.txt, consisting of all the parameters described above:$n, k, \lambda 1, \lambda 2$. A sample input file is: 100 100 5 20.

**Output:** Your program should output to a file in the format given in the pseudocode for each algorithm. A sample output is for CLH lock is as follows:

CLH lock Output:
1st CS Requested Entry at 10:00 by thread 1
1st CS Entered at 10:05 by thread 1
1st CS Requested Exit at 10:06 by thread 1
1st CS Exited at 10:06 by thread 1
1st CS Requested Entry at 10:01 by thread 2
.
.
.

Similar to CLH lock, you have to show the output for TAS, TTAS and MCS Locks as well. The output must demonstrate the mutually exclusive execution of the threads.

**Report:** You have to submit a report for this assignment. This report should contain a comparison of the performance of both locking algorithms. You must run both these algorithms multiple times to compare the performances and display the result in form of a graph. You must average each point in the graph plot by 5 times.

You run both these algorithms varying the number of threads from 2 to 64 in the powers of 2 while keeping other parameters same. Please have $k$, the number of CS requests by each thread, fixed to 10 in all these experiments while having $\lambda 1, \lambda 2$ as 1 and 2.
The graphs in the report will be as follows for each algorithm:

- the x-axis will vary the number of threads from 2 to 64 in the powers of 2 (as explained above).
- the y-axis will show (1) csEnterTime: the average time taken to enter the CS by each thread and (2) csExitTime: the average time taken to exit the CS by each thread.

It must be noted that for each algorithm, TAS, TTAS, CLH and MCS, there will be two curves: entry and exit times. Thus, there will be eight curves for the four locking algorithms.

Finally, you must also give an analysis of the results while explaining any anomalies observed in the report.

**Deliverables:** You have to submit the following:

- The source files containing the actual program to execute. Please name them as: TAS-<rollno>.cpp, TTAS-<rollno>.cpp, CLH-<rollno>.cpp, MCS-<rollno>.cpp.
- A readme.txt that explains how to execute the program
- The report as explained above

Zip the two files and name it as ProgAssn5-<rollno>.zip. Then upload it on the google classroom page of this course. Submit it by the above mentioned deadline.