# Parallel and MultiThreaded Programming

# CSYE 7215

# Homework 10

# Due: April 5, 2020

Put all your java, compiled class files and documentation into zip file named Homework10.zip and submit it via the dropbox on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this assignment.

1. Describe:

Blocking Algorithms?

Non-Blocking Algorithms?

What are Intrinsic Lock and Explicit Lock?

What Structure do Intrinsic locks use? blocking or non-blocking? Why?

Compare and Swap, how does it work? Provide hardware level description.

Compare and Set, how does it work?

What is Contention Detection?

With Atomic operations, how does contentions are detected by Threads?

When a Thread fails in an update operation, does it really fail? Does it cause for

all run time operations to fail? Yes/No, explain.

ReentrantLock provides same locking as intrinsic locking? Ture/False? Why?

Intrinsic locks have significant advantages over explicit locks, True/False?

Why exclusive locking is a pessimistic technique?

What is an Optimistic approach? Explain

Explain how Java Atomic operations (eg: AtomicInteger, AtomicReference)

relates to hardware support for concurrency?

Lock-based algorithms are at risk for a number of liveness failures. If a thread

holding a lock is delayed due to blocking I/O, page fault, or other delay, it is

possible that no thread will make progress. Why is this True?

2. What are the difference between Condition interface and Atomic operation, namely blocking and non-blocking algorithms. Explain in details.

3. How would you use AtomicReference. Give example in detail.

4. How does C++ condition\_variable library works, the purpose?

How do you compare with Java?

5. What are differences between “Condition queue” and “Condition”?

Consider the following Condition interface to answer the question.

public interface Condition {

void await() throws InterruptedException;

boolean await(long time, TimeUnit unit) throws InterruptedException;

long awaitNanos(long nanosTimeout) throws InterruptedException;

void awaitUninterruptibly();

boolean awaitUntil(Date deadline) throws InterruptedException;

void signal();

void signalAll();

}

6. This is a Non-blocking Counter Using CAS. How does it work?

@ThreadSafe

public class CasCounter {

private SimulatedCAS value;

public int getValue() {

return value.get();

}

public int increment() {

int v;

do {

v = value.get();

} while (v != value.compareAndSwap(v, v + 1));

return v + 1;

}

}

7. A Non-Blocking Stack is implemented using Stack and LinkList data structures as described. The code uses AtomicReference to construct a Stack.

a) Explain how the construction of Stack works using AtomicReference?

b) How does top.compareAndSet operation works in stack construction?

Provide insight at the hardware processor level?

c) Write the code for Non-blocking Stack. How do you manage Atomic operations?

Compile and Run

8. A Linked queue is more complicated than a stack because it must support fast access to both the head and the tail. It maintains separate head and tail pointers.

a) How does the two pointers are managed with CAS operations?

b) When you insert a new element, both pointers are updated atomically. How?

c) Do you use two CAS operations, one for each pointer update? Does it work?

d) You need to consider different Tricks/States to Insert new element in queue.

What are those tricks and states? How does it work? Explain

e) Write the code for Non-blocking Queue. How do you manage Atomic operations?

Compile and Run