COP 4520 Spring 2019

Homework Assignment 1

Note 1:

This assignment is individual.

Please, submit your work via Webcourses.

Submissions by e-mail will not be accepted.

Due date: Wednesday, February 6th by 11:59 PM.

Late submissions are not accepted.

You are given the partial pseudocode for a concurrent stack class using locks. In this design, *head* points to the node at the top of the stack, and *numOps* refers to the number of operations that have occurred during a given execution. The Node class contains a generic value and a pointer to the next node in the stack. The constructor for the stack class allocates a lock and sets the head to NULL, indicating the stack is empty.

The push operation first acquires the lock. If this is successful, a node *n* is allocated with the given value *x*. *n*'s next pointer is then directed at the current head of the stack. Afterwards, n is made the new head of the stack and *numOps* is incremented.

The pop operation is similar to the push operation. First, it acquires the lock and checks if the stack is empty. If the stack is empty, it releases the lock and returns *null*. Otherwise, it retrieves the node from the top of the stack, changes *head* to *n.next*, and *numOps* is incremented.

Your assignment is as follows:

1. Remove the lock element, and modify the push and pop operations using atomic operations to transform the stack into a lock-free stack. Briefly discuss the correctness (i.e. linearizability) and progress (i.e. lock-freedom) properties of your modified lock-free stack.
2. Observe the variables in the stack class and decide which of them should be declared **volatile (or atomic)**. Consider the atomics library functionalities of the programming language of your choice. Find out whether an atomic variable is by default volatile or not. Please explain. Discuss what could go wrong if you run your program without declaring these variables **atomic** or **volatile.** Be careful not to declare variables atomic/volatile if this is not necessary as this would lead to significant performance loss.

Submission Instructions:

You can use a programming language of your choice for this assignment.

Please submit a compressed folder containing your implementation and a brief README file.

Additional Instructions:

Cheating in any form will not be tolerated.

```
public class Stack<T>{
    Lock lock
    Node* head
    int numOps = 0

    class Node {
        T val
        Node* next

        Node(T _val) {
            val = _val
        }
    }
    public Stack() {
        lock = new ReentrantLock()
        head = NULL
    }
    public bool push(T x) {
        lock.lock()
        Node *n = new Node(x)
        n.next = head
        head = n
        numOps++
        lock.unlock()
        return true
    }
    public T pop() {
        lock.lock()
        if head == NULL then
            lock.unlock()
            return NULL
        Node *n = head
        head = n.next
        numOps++
        lock.unlock()
        return n.val
    }
    public int getNumOps() {
        return numOps
    }
}
```