HY486: Principles of Distributed Systems
Professor: Panagiota Fatourou

1st Set of Exercises: Theory Part

***Deadline**: March 28, 2016*

Exercise A [25%]

A parking place on an island has capacity N, i.e. at most N cars can park in it. A boat with capacity M < N cars performs trips from the island to the mainland and vice versa. The boat leaves from each port either if it is full or if there are no cars waiting at the port of departure but there are cars waiting at the destination port.

1. Solve this synchronization problem using locks. Your solution must avoid the problems of deadlock and starvation.
2. Provide an informal intuitive description of your implementation.
3. Argue that your algorithm is correct.
4. Argue about the progress properties of your implementation.
5. Analyze the complexity of your algorithm on a cache-coherent NUMA architecture and a cache-less NUMA machine. To do so, choose a specific lock implementation, and provide bounds on the number of cache misses that occur on a cache-coherent SMP architecture, as well as on the number of remote memory references that occur on a cache-less NUMA architecture.

Exercise B [25%]

1. Consider the unbounded lock-free queue implementation. Explain (with a counter-example) why the order of lines 29 and 30 (page 8 - Section 5: Pools) is important for correctness.
2. Read Section 7.6 of Chapter 7, M. Herlihy and N. Shavit, "The art of multiprocessor programming".
    a. Present C-like pseudo-code for the TOLock described in this section.
    b. Explain, in your own words, the way the algorithm works and why you think it is correct.
    c. Discuss the liveness properties of the algorithm, the number of cache-misses that it causes, and its space complexity.

Exercise C [50%]

In this exercise, you have to design and analyze a **lock-based** implementation of a concurrent binary search tree. In your implementation, you should use a lock per node and have each operation (BSTInsert, BSTDelete, and BSTSearch) holding locks for at most a constant number of nodes at each point in time. Moreover, operations occurring in different sub-trees should occur concurrently (so, the use of a single global lock is not allowed). Your implementation should use fine-grain locking in order to increase parallelism.

1. Present pseudocode for BSTInsert, BSTDelete, and BSTSearch for your implementation.
2. Provide informal intuitive descriptions of your implementation.
3. Fix any execution of your implementatio and explain how linearization points should be assigned to the operations of each execution.
4. Argue that your algorithms are correct.
5. Argue about the progress properties of your implementation.
6. Analyze the complexity of your algorithms on a cache-coherent NUMA architecture and a cache-less NUMA machine. To do so, provide bounds on the number of cache misses that occur on a cache-coherent SMP architecture, as well as on the number of remote memory references that occur on a cache-less NUMA architecture.