Chad Weirick

CST-221

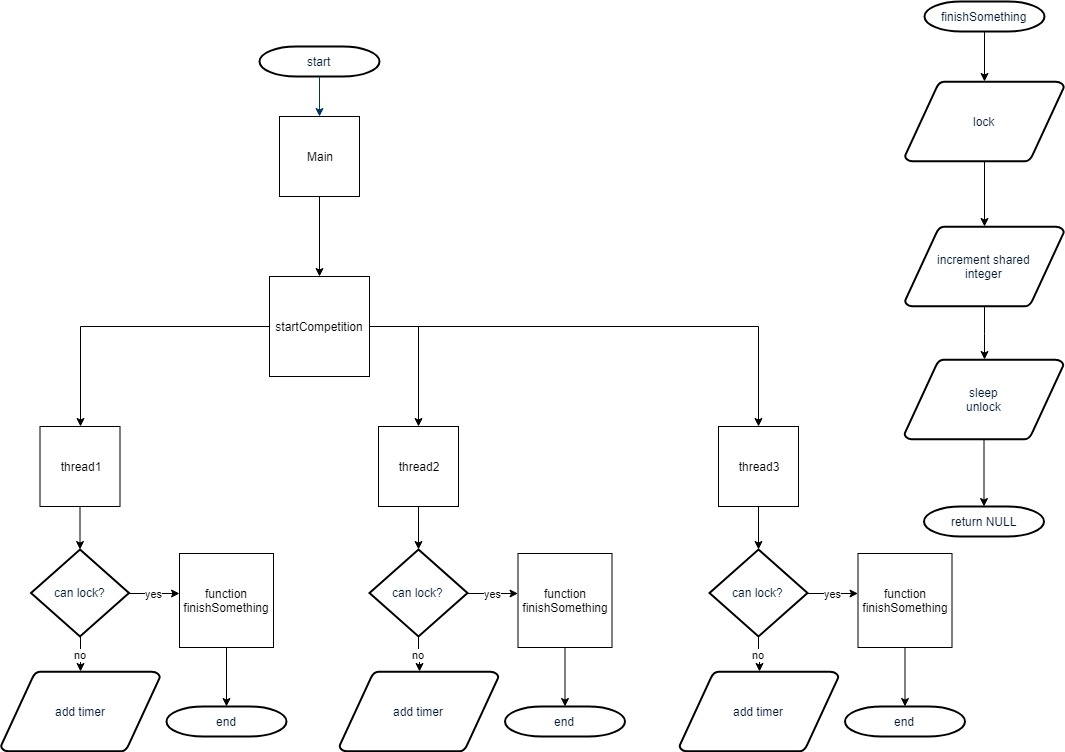
Deadlock Avoidance

**Scenario Description**

In this scenario there will be multiple threads attempting to access a single integer protected with a mutex system. If a process is unable to gain lock/access, it will wait, and try again. In order to simulate a longer lock, sleep has been included in the locked portion accessing the shared integer.

**Flowchart**

The flowchart is also included in the files, but this will include a brief summary:



When the program is run, it creates multiple threads. Each thread will determine if it can lock the integer. If it can, it will start the finishSomething function. FinishSomething will lock the variable, sleep briefly to simulate a longer process that could be locked, increment the variable to do measurable work, and then unlock and return a NULL value.

**Considering Suitability**

Monitors on the other hand are designed to queue requests to specific resources. This means that if a request comes up for a free resource, it can be processed without waiting. If the resource is busy, it is queued for that resource and that resource only. This prevents performance issues, but also makes it far less likely that any of the above conditions could be backed up due to the volume of transactions.

The downside of monitors appears to be slightly greater resource utilization per use as they are an abstract data type. The results on a large scale like a banking system could result in higher operating costs, greater IT budgetary requirements, and additional data center infrastructure in terms of both cooling and power delivery.

**Challenges**

I believe that I have run into a logic issue but have not been able to track this down. I believe that there is an edge condition causing multiple threads to compete and be similarly time-locked. If this is the case, potential fixes might include adding a base plus random time element to ensure that such a situation never happens. Your feedback would be appreciated.

**GitHub Information**

The GitHub link is as follows: https://github.com/chadatgcu/threading.git