Brandon Horner
Michael Gowanlock
CS499: Parallel Programming
23 September 2019

Assignment 2: Introduction to Pthreads

1) False
    - Since either of the two threads could start first, we could see the two results:
      Example 1 -- thread1 then thread2
        - ➢ Starting1
        - ➢ Exiting1
        - ➢ Starting2
        - ➢ Exiting2

      Example 2 -- thread2 then thread1
        - ➢ Starting2
        - ➢ Exiting2
        - ➢ Starting1
        - ➢ Exiting1
    - The two threads could run for a small amount of time each. Possibly printing one line in thread1, then thread2 prints completely, then thread1 finishes execution afterward:
      Example 3
        - ➢ Starting1
        - ➢ Starting2
        - ➢ Exiting2
        - ➢ Exiting1

2) False
    - Since either of the two threads could completely execute the do_work() function first, we could see the two results:
      Example 1 -- thread1 then thread2
        - ➢ 5
        - ➢ 12

      Example 2 -- thread2 then thread1
        - ➢ 12
        - ➢ 5

3) False
    - Starting with either thread executing the do_work() function, if one thread finishes before the other, the sum will be updated to 3 or 5 first, then updated to 8, finally printing 8.

Example 1

- ➢ Sum: 8

- Now in the off-chance that both threads try to access and change the value of count concurrently, we have a race condition.

  Example 2 - thread1 reads in sum at the same time as thread2, then updates sum after thread2

  - ➢ Sum: 3

  Example 3 - thread2 updates sum second, thread1's change is overwritten

  - ➢ Sum: 5

4) False
   - Either thread1 or thread2 could enter the critical section first in the do_work() function, so the count that will be printed will be either:

     Example 1 -- thread1 enters the critical section first
     - ➢ 2
     - ➢ 5

     Example 2 -- thread2 enters the critical section first
     - ➢ 3
     - ➢ 5
5) False
   - Since both threads are using different mutex locks, neither one is held up by the other. Therefore we could have the following possibilities for output:

     Example 1 -- thread1 enters and completes the "critical section" first
     - ➢ 2
     - ➢ 5

     Example 2 -- thread2 enters and completes the "critical section" first
     - ➢ 3
     - ➢ 5
   - If both threads were in the critical section at the same time, they could both update the value of count before any printing is done, resulting in:

     Example 3
     - ➢ 5
     - ➢ 5