Name: Shubham Sharma

Section: L

University ID: 866653811

**Lab Report**

**Summary:**

**3.2:**

1. The two threads increment args a million times each. Hence the value of ‘shared’ should be ‘2000000’. So the expected output is:

After both threads are done executing, `shared` = 2000000

1. The output on running the program is:

After both threads are done executing, `shared` = 1023748

1. The two threads were trying to access the same resource at the same time, hence causing one resource to hold and the other to wait while iterating a million times. The reason it is not 2 million but around 1 million is because, there is about a 50/50 chance of one thread getting access to the variable and the other waiting. Hence, only half of the iterations actually increment the ‘shared’ value.

**3.3:**

1. Yes, this approach fixed the issue in the original code. As the output provided a result same as the expected result. This is due the thread synchronization added in on the critical section of the code where the shared variable is being accessed. The new output is shown below:

After both threads are done executing, `shared` = 2000000

**3.5:**

1. The minimum number of conditions needed for the example to work as intended is 3 conditions.
2. The conditions are as follows: // TODO
   1. There will need to be a signal to show that the producer has relented and C1 can lock in.
   2. Then, there will need to be a signal to show that C1 has relented and C2 can lock in.
   3. Then, there will need to be a signal to show that C2 has relented and P can produce again.