Name: Shubham Sharma
Section: L
University ID: 866653811

Lab 4 Report

Summary:

This lab we learnt about the uses of thread synchronization and how to implement them. We did this my utilizing mutex locks and understanding critical sections and the uses of conditional variables for different threads.

The we move on to implementing a working printer server which is capable of handling and printing jobs accordingly.

3.2:

A. 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
```

B. The output on running the program is:

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

C. 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:

A. 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:

A. The minimum number of conditions needed for the example to work as intended is 3 conditions.
B. The conditions are as follows:
    a. A condition signal to show the producer code that has completed, hence C2 code can utilize it.

b. A condition signal to show that C1 code has completed and C2 code can utilize the code.
c. A condition signal signal to show that C2 code has completed and hence the producer can utilize it.

DISCLAIMER: Could not complete the rest.☹