Student Name: - ATHARVA DESHPANDE

Student Email: - deshpana@oregonstate.edu

Project No: - Project#3

Project Name: - Mutex Stack Challenge

ATHARVA DESHPANDE **CS575** **deshpana@oregonstate.edu**

1. Tell what machine you ran this on.
   ➔ I ran the program on a Windows 11 machine using the OSU Flip1 server. I wrote the code using vim and executed it on the server to get the results with and without mutex locks.

2. Tell what operating system you were using.
   ➔ I was using the Linux Centos release 7.9.2009 Operating system on the OSU flip1 server.

3. Tell what compiler you used.
   ➔ The complier used was g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44).

4. Include, in your writeup, the pieces of code where you implemented the mutexes.
   ➔ I included them in the push and pull functions. The mutex should be locked very late into the code and released as soon as possible to improve performance. It is the only part of the entire code that would benefit from mutex locks as both functions perform direct operations on the same stack.

5. Tell us what you discovered by doing this:

a. Does the non-mutex way of doing this ever work? If so, how often?
   ➔ For the execution results that I recorded, there were no instances where even when mutex was not used, the number of pop errors was 0. It may occur in rare cases of coincidence when the threads do not share a mutual resource (stack in this reference).

b. Does changing NUMN make any difference in the failure percentage?
   ➔ Though there were more pop errors by changing the NUMN, the failure percentage remained in the same range for all trials. This range was observed to be 16-59%. There were only two instances where the failure percentage went above 60%. For all the trials where the mutex was on, the failure percentage remained 0 even with different value of NUMN.

c. Is there a difference in elapsed execution time between mutex and non-mutex? Why do you suppose this is? (Ignore the very large, elapsed times -- these are a result of the TIMEOUT being used up.)
   ➔ Initially there is a small difference between elapsed execution time between mutex and non-mutex executions. But as we increase the number of threads, the difference between elapsed execution time increases as well. Further, we can see that the elapsed time for executions where mutex was on was more than that when mutex was off. This is because mutex introduces additional overhead while code execution which may impact performance.