LAB REPORT- ASSIGNMENT-1

SUBMITTED BY: - Parth Parashar

I found this assignment to be extremely helpful for understanding the thread support for these languages. I had a pleasant experience with the assignment

1) All the programs are running without any exception or errors. All the programs pass all the tests given in the run scripts as well as they pass all the conditions mentioned in the

lab assignment

2) The programs are exhibiting parallelization behaviour and are working as expected. I do see the interleaving execution among the threads. The threads are working independently until

and unless specified by the condition of wait and notify/notifyAll. For multi-consumer programs, the workload is evenly distributed among the threads and each thread is performing at-least one task of removal in case of consumers and addition in case of producers.

3) I had a very pleasant experience with all the programs. I have been coding in java for quite a long time so switching back to C++ was a bit challenging as I had to lookup the syntax

again and again. Also, the use of locks can be a bit confusing at times. This might be because I have not coded in C++ in a very long time.

one of the most major issue that I had to face was in case of multiple consumers for program-2. There, when I first ran the program, it was behaving sequentially and only one consumer was working. The rest were not working. They were idle.

This was because I had put the mutex lock at the wrong place and was using notify instead of notify all. This lead to a siituation where I was not able to figure out what to do and how

to approach the problem. I did not encounter a similar problem in java because of the use of synchronized block which takes care of the locking mechanism and hence the only thing that

I had to worry about was the proper use of notify and wait. But in C++, lock and unlock plays a major role and hence I had to devote a lot of time on it. Same goes for the third program

for support of multiple producers and multiple consumers in C++. I again had to figure out the correct positioning of the lock

One more area where I had to dig deep to resolve the issue was the multiple producer and multiple consumer problem. I had to figure out a way for how to divide the segments.

For the Java programs, while the code writing was smooth, the one area where I was stuck was the use of runj2 and runj3 script which were not able to figure out my main class.

This was because I was using packages in it. As I removed the package statement, it started working with the script but I had to spend a lot of time trying to figure out the issue in itself.

As for the support of threads in these languages, while java is easier to use, C++ seems to be more flexible and gives more freedom to play around with locks and unlocks

For me, Java is the better one though because it appears a lot more cleaner and more refined. The threads are handled better by the JVM and all-in-all is better in terms of outright usability.