Assignment – 1 Report

Name: Deepa Subray Hegde
PSU ID: 954680127

1. Status of your programs. Do they pass all tests? If not, describe the remaining issues as clearly as possible.

   Yes

2. Parallelization behavior. Do you see interleaving execution among all the threads? For multi-consumer programs, are the workload evenly distributed among the threads?

   Yes, I do see interleaving execution among the threads. However, in the Java sometimes there is no interleaving. Producer runs continuously until the queue is full adding 20 items and then the consumer runs continuously removing 20 items until the queue is empty.

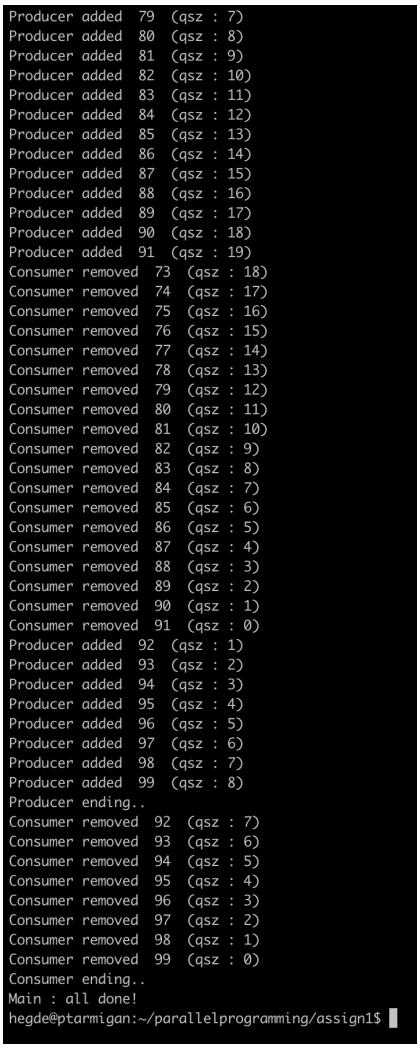
   The workload in C++ is distributed almost evenly but in Java the load balancing is not as evenly distributed as C++. There are some of the threads which don't get to consume any item

3. Experience and lessons. What issues did you encounter? How did you resolve them? What is your impression of the two languages' thread support?

   Base Version:
   C++ program passes all the tests – Synchronization was smooth

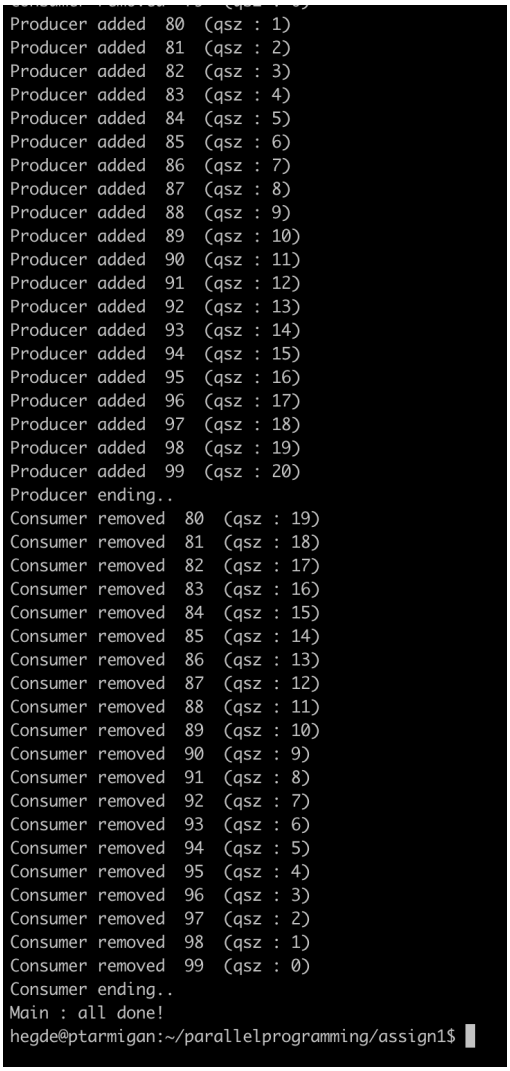
   Java program – The 2 consumer and producer thread are working. However, there are 2 variants of outputs I am seeing

   Sample 1:



The producer and consumer threads interleave

Sample 2:



As shown above, the producers run and add 20 items to the queue (until queue is full) and then the consumer starts consuming the items until the queue is empty.

I have added 2 synchronized blocks in the producer. One surrounding the wait and one surrounding the q.add() and notify().
One synchronized block for consumer which surrounds the wait and removing logic.

Extended Version:

Cpp – passes all tests and synchronization was smooth again. Load balancing between the consumer threads works fine.

Java – passes all tests. Load balancing is not even. Not all consumers get a chance to consume.

```
Consumer Stats: [ 60, 20, 0, 0, 20, ] Total = 100
```

```
Consumer Stats: [ 22, 0, 0, 40, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24, ] Total = 100
```

Just like the previous java program, I have added 2 sync blocks for producer. One surrounding the wait and one surrounding the q.add() and notify().
One synchronized block for consumer which surrounds the wait and removing logic.

```
Consumer [3] removed  82  (qsz : 9)
Consumer [3] removed  83  (qsz : 8)
Consumer [3] removed  84  (qsz : 7)
Consumer [3] removed  85  (qsz : 6)
Consumer [3] removed  86  (qsz : 5)
Consumer [3] removed  87  (qsz : 4)
Consumer [3] removed  88  (qsz : 3)
Consumer [3] removed  89  (qsz : 2)
Consumer [3] removed  90  (qsz : 1)
Consumer [3] removed  91  (qsz : 0)
Producer added  92  (qsz : 1)
Producer added  93  (qsz : 2)
Producer added  94  (qsz : 3)
Producer added  95  (qsz : 4)
Producer added  96  (qsz : 5)
Producer added  97  (qsz : 6)
Producer added  98  (qsz : 7)
Producer added  99  (qsz : 8)
Consumer [14] removed  92  (qsz : 9)
Consumer [14] removed  93  (qsz : 8)
Consumer [14] removed  94  (qsz : 7)
Consumer [14] removed  95  (qsz : 6)
Consumer [14] removed  96  (qsz : 5)
Consumer [14] removed  97  (qsz : 4)
Consumer [14] removed  98  (qsz : 3)
Consumer [14] removed  99  (qsz : 2)
Consumer [14] removed  -1 and ending..  (qsz : 1)
Consumer [13] removed  -1 and ending..  (qsz : 0)
Consumer [0] removed  -1 and ending..  (qsz : 4)
Consumer [1] removed  -1 and ending..  (qsz : 3)
Consumer [2] removed  -1 and ending..  (qsz : 2)
Consumer [3] removed  -1 and ending..  (qsz : 1)
Producer ending..
Consumer [4] removed  -1 and ending..  (qsz : 8)
Consumer [5] removed  -1 and ending..  (qsz : 7)
Consumer [6] removed  -1 and ending..  (qsz : 6)
Consumer [7] removed  -1 and ending..  (qsz : 5)
Consumer [8] removed  -1 and ending..  (qsz : 4)
Consumer [9] removed  -1 and ending..  (qsz : 3)
Consumer [10] removed  -1 and ending..  (qsz : 2)
Consumer [11] removed  -1 and ending..  (qsz : 1)
Consumer [12] removed  -1 and ending..  (qsz : 0)
Consumer Stats: [ 22, 0, 0, 40, 14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 24, ] Total = 100
Main : all done!
```

The threads are parallelized. But as mentioned in the previous version, sometimes the producers keep on adding until the queue is full and then the consumers start consuming continuously until the queue is empty

Extended Version:

Cpp – Passed all tests. Load balancing works fine

Observation from this assignment: Even though java looks simple but is very tricky and time consuming to debug and synchronize using the sync blocks. Also, the load balancing is better in C++ than in Java as per my observation