UNIVERSITY OF VIRGINIA, DEPARTMENT OF COMPUTER SCIENCE

Lab 06 - Concurrency

Dr. Mark R. Floryan

September 6, 2018

1 PRE-LAB

This week, you will be implementing a simple concurrent queue. For this assignment, we are going to focus on correctness (i.e., the queue must work with multiple threads without crashing). Optimizing the speed of your concurrent queue is optional.

1. Download the provided starter code
2. Implement the ConcurrentQueue class
3. Run the main in MainTester.java to ensure your queue works correctly
4. **FILES TO DOWNLOAD:** concurrency.zip
5. **FILES TO SUBMIT:** lab06.zip

1.1 CONCURRENTQUEUE.JAVA

To begin, implement the *ConcurrentQueue* class inside the ConcurrentQueue.java file. The methods you are responsible for are listed below. This Queue **must be a linked-list based queue**. You may use Java's built-in Linked List (import java.util.LinkedList) or you may use your own implementation from the previous labs.

```
public class Queue<T>{
    public Queue();

    public int size();

    public void enqueue(T data);

    public T dequeue();
}
```

Your class must be alterable by different threads. Thus, you should protect the fields (head, tail, etc.) of your linked list from being corrupted by multiple threads accessing the queue at once. For this lab, we care about *correctness* (i.e., the queue works as intended with multiple threads even if it could be optimized to run faster).

1.2 MAINTESTER.JAVA

You can test your code by running the main method in *MainTester.java*. This tester will first test your queue using a single thread and time the results. Then, the method will test your queue again using two threads and time the results again. Any errors that occur should be printed to the console. Make sure you run the tester multiple times. Race conditions can sometimes cause code to appear to work but not consistently.

1.3 OPTIONAL: OPTIMIZING

Although we only care about correctness for this lab, if you are interested you should try to make your queue run as quickly as possible. What can you do to increase the degree to which multiple threads can use the queue in parallel? How fast can you make the concurrent test that we provided?

When you are done, submit your entire project as a zip file to Collab.

2 In-Lab

The goal of this in-lab is to continue practicing with concurrent programming in a laid back environment. As usual, you will:

1. Get into small groups of two

2. The TAs will present you with some programming challenges regarding threads. These will not be graded, but you are required to attend and to participate.

3. You may think about the challenges before lab below if you'd like, but we highly recommend that you not solve them ahead of time.

4. The TAs will give you time to solve each problem and lead you in sharing solutions with one another.

2.1 Concurrency Coding Challenges

The TAs will lead you in going through the following challenges. It is ok if you do not get through each of these.

1. Write a program that performs matrix multiplication in parallel. If you tweak the number of threads and what each thread does, How fast can you make the program run?

2. You want to write a program that accepts a list of numbers and filters out all of the values that are outside of a certain range. Write a multi-threaded program that does this. How fast can you make the code run?

3. Write a threaded program that simulates a rock, paper, scissors tournament. Given a number of competitors, pair them off and have each compete (randomly select RPS for each and see who wins). Then pair off the winners and repeat until you have a champion. Your competitors will simply be numbered 1 through n. Use multithreading to speed up your program. How fast can you make your code run?

If you have coded up all three of these and they work, then show the TA and you may leave early.

3 POST-LAB

The goal of this post-lab is to write a report analyzing the efficiency of your concurrent queue from the pre-lab. Specifically, we are interested in how efficient your queue is compared to a sequential queue as you increase the number of items being added / removed and as you increase the number of threads accessing the queue in parallel.

You will perform an experiment by doing the following:

1. Write some test code (or use/modify what we provided) so that you can test the efficiency of your queue sequentially and concurrently.

2. Run a small experiment where you increase the number of elements added/removed to the queue until the queues start to become noticeably slow. Time how long your sequential vs concurrent queue took to complete.

3. Do a second experiment where you increase the number of threads accessing the concurrent queue at once. Time how long each test case takes to complete. For this test, keep the number of insertions / deletes constant.

4. Write a report summarizing and analyzing your findings

5. **FILES TO DOWNLOAD:** None

6. **FILES TO SUBMIT:** PostLabSix.pdf

3.1 REPORT

Summarize your experiment and your findings in a report. Make sure to adhere to these general guidelines:

- Your submission MUST BE a pdf document. You will receive a zero if it is not.

- Your document MUST be presented as if submitted to a professional publication outlet. You can use the template posted in the course repository or follow Springer's guidelines for conference proceedings.

- You should write your report as if it is original novel research.

- The grammar / spelling / professionalism of this document should be sound.

- When possible, do not use the first person. Instead of "I ran the code 60 times", use "The code was executed 60 times...".

In addition to the general guidelines above, please follow the following rough outline for your paper:

- **Abstract**: Summarize the entire document in a single paragraph
- **Introduction**: Present the problem, and provide details regarding the two strategies you implemented.
- **Methods**: Describe your methodology for collecting data. How many method calls, how many executions, how you averaged things, etc.
- **Results**: Describe your results from your execution runs.
- **Conclusion**: Interpret your results. What happens as inserts / deletes in increased? What about number of threads? Did anything about the times you reported seem odd? If so, why do you think that might be?

Lastly, your paper MUST contain the following things:

- A table (methods section) summarizing the experiments and how many execution runs were done in each group.
- A table (results section) summarizing the results of experiment 1 (number of inserts / deletes).
- A table (results section) summarizing the results of experiment 2 (number of threads).
- Some kind of graph visualizing the results of the table from the previous bullet.