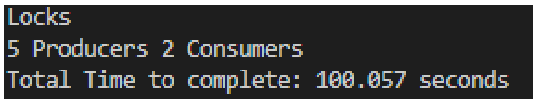
Kyle Kirtland

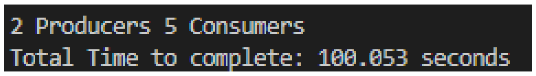
Homework 5

https://github.com/Redster11/CS_3700/tree/master/HW5/Producer%20Consumer

Producer Consumer

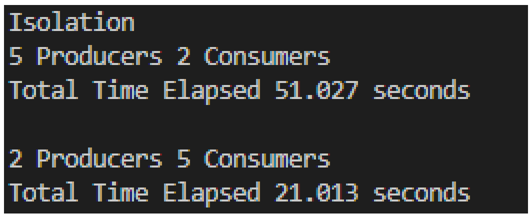


```
Locks
5 Producers 2 Consumers
Total Time to complete: 100.057 seconds
```



```
2 Producers 5 Consumers
Total Time to complete: 100.053 seconds
```

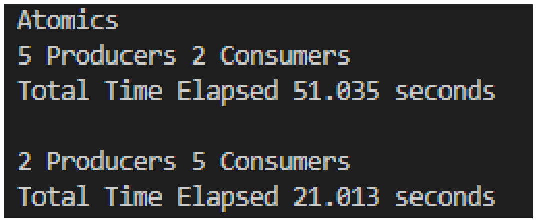
Locks: Due to only one item being able to work at a time it takes the total time of the number of items



```
Isolation
5 Producers 2 Consumers
Total Time Elapsed 51.027 seconds

2 Producers 5 Consumers
Total Time Elapsed 21.013 seconds
```

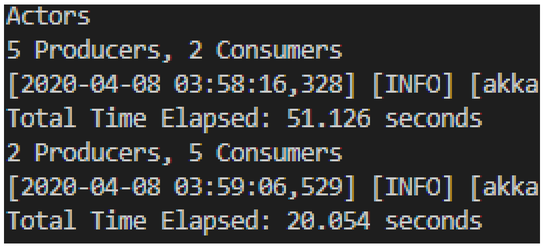
Isolation: since the only part that has a time to it is the consumers the time taken is directly proportional to the number of consumers.



```
Atomics
5 Producers 2 Consumers
Total Time Elapsed 51.035 seconds

2 Producers 5 Consumers
Total Time Elapsed 21.013 seconds
```

Atomics: since the only part that has a time to it is the consumers the time taken is directly proportional to the number of consumers.



```
Actors
5 Producers, 2 Consumers
[2020-04-08 03:58:16,328] [INFO] [akka
Total Time Elapsed: 51.126 seconds
2 Producers, 5 Consumers
[2020-04-08 03:59:06,529] [INFO] [akka
Total Time Elapsed: 20.054 seconds
```

Actors: since the only part that has a time to it is the consumers the time taken is directly proportional to the number of consumers.

Findings:

I found that locks are very slow, and all the other ways are about the same other than actors where they have almost a full second faster runtime on the 2 Producer and 5 Consumer.

Overall, the projects follow the idea of time = numberToProduce /consumerAmount in seconds. This is because it takes each consumer 1 second to complete one task meaning that it is the lowest part of our program. Of course, there is a bit of overhead which causes the actual time elapsed to be longer than the expected number however the numbers are very close to expected.

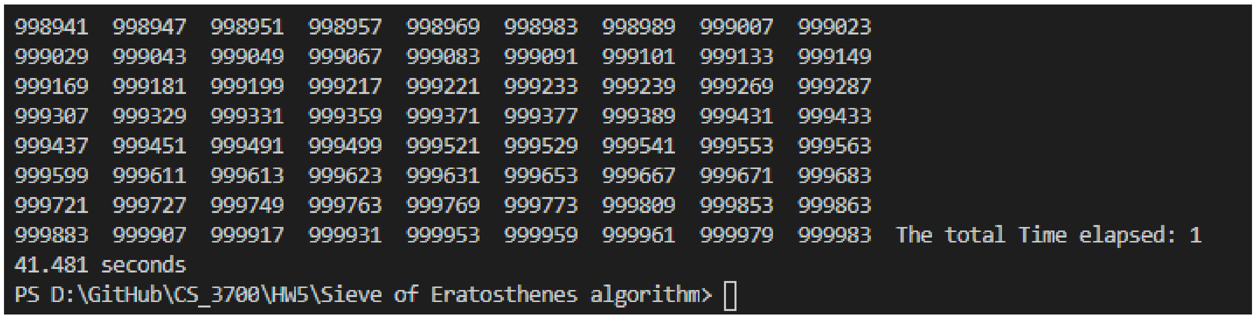
Actors seem to be able to get faster times depending on the actions that they are performing, the consumers seem to get to almost actual time expectation on the run of 2 Producers and 5 consumers. This may also just have been a coincidence in the system.

Kyle Kirtland

https://github.com/Redster11/CS_3700/tree/master/HW5/Sieve%20of%20Eratosthenes%20algorithm

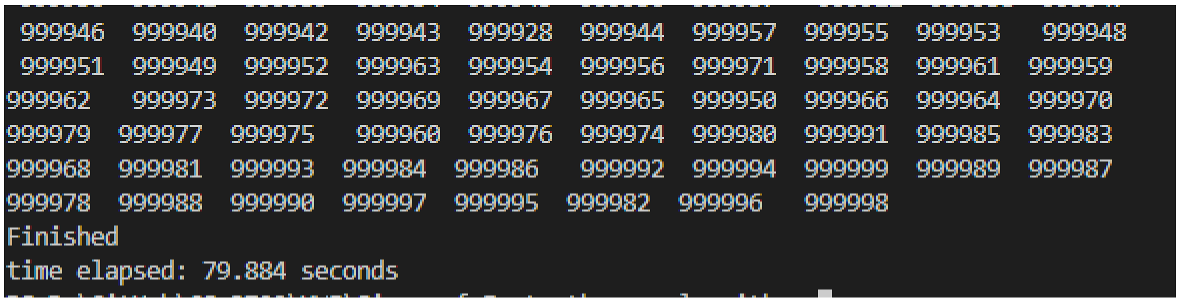
The Sieve of Eratosthenes

Single Threaded



```
998941  998947  998951  998957  998969  998983  998989  999007  999023
999029  999043  999049  999067  999083  999091  999101  999133  999149
999169  999181  999199  999217  999221  999233  999239  999269  999287
999307  999329  999331  999359  999371  999377  999389  999431  999433
999437  999451  999491  999499  999521  999529  999541  999553  999563
999599  999611  999613  999623  999631  999653  999667  999671  999683
999721  999727  999749  999763  999769  999773  999809  999853  999863
999883  999907  999917  999931  999953  999959  999961  999979  999983  The total Time elapsed: 1
41.481 seconds
PS D:\GitHub\CS_3700\HW5\Sieve of Eratosthenes algorithm> 
```

Actors



```
 999946  999940  999942  999943  999928  999944  999957  999955  999953   999948
 999951  999949  999952  999963  999954  999956  999971  999958  999961  999959
999962   999973  999972  999969  999967  999965  999950  999966  999964  999970
999979  999977  999975   999960  999976  999974  999980  999991  999985  999983
999968  999981  999993  999984  999986   999992  999994  999999  999989  999987
999978  999988  999990  999997  999995  999982  999996   999998
Finished
time elapsed: 79.884 seconds
```

Findings:

Time for Single threaded action was 141.481 seconds whereas the speed for Actors was 79.884 seconds. The speedup of the program was 1.77 times faster than the single threaded program. Considering that we are only working with 2 threads in this program, the maximum speedup would be around 2 times faster.

Actors seem to be easier to work with since they have many of the functions already build into their code. This makes programming them much easier to use.