Zack Campbell: zcc254
Paul Heath: pah2276

Homework 2

1. Int rename(int j):

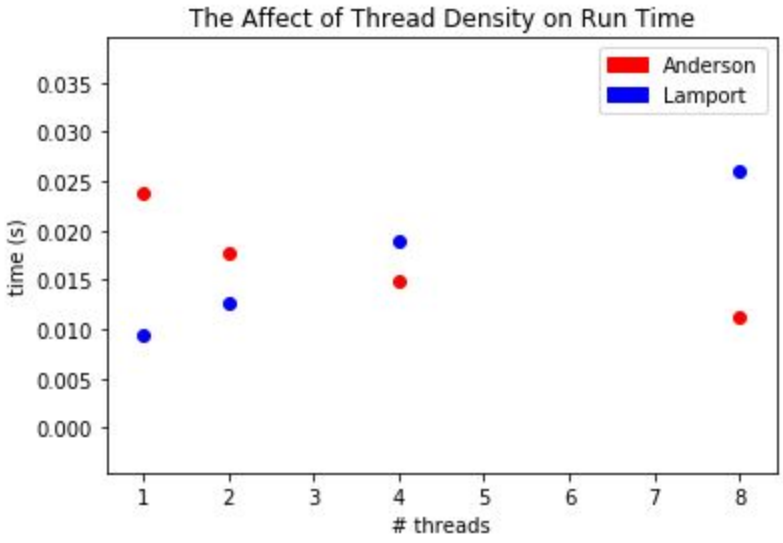
```
//shared variables
Int M = (n * (n + 1)) / 2 // where n is the number of threads to rename
Int newIdx // initialized to zero but not on each function call

//private variable
Int ret

LamportFastMutex.acquire(j) // as defined with splitters in textbook chapter 2
++newIdx
Ret = newIdx
LamportFastMutex.release(j) // as defined with splitters in textbook chapter 2

return ret;
```

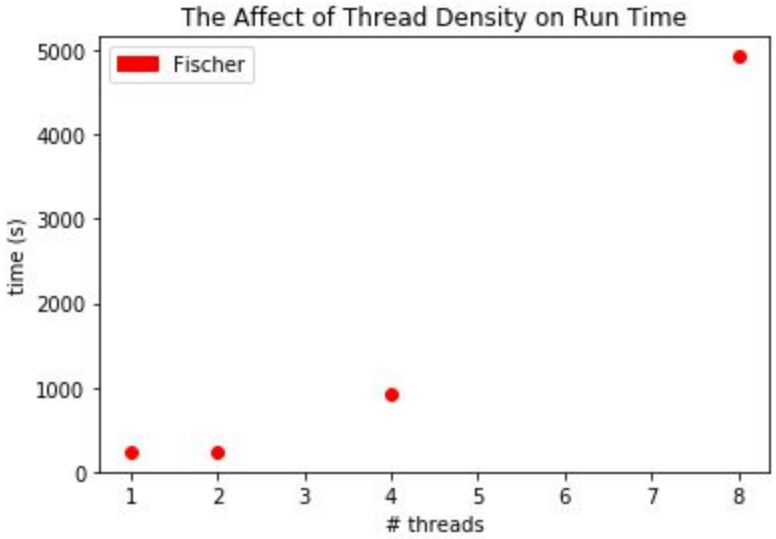
LamportFastMutex:

```
Int X, Y // initialized to -1
bool flag[1 to n] //

acquire(int i):
    // the code for this is in the textbook, so I don't want to copy it verbatim
    // this uses splitter

release(int i):
    // the code for this is in the textbook, so I don't want to copy it verbatim
```

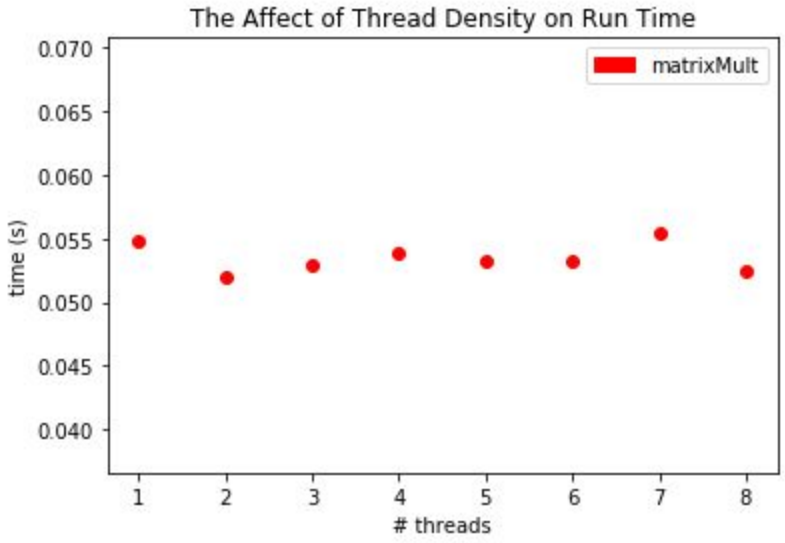
Essentially, use Lamport's Fast Mutex Algorithm to secure a variable that you increment. Lamport's Fast Mutex Algorithm uses a splitter to protect the secured variable.
Each call of rename will acquire the lock, increment the shared variable and get the next unique index from the shared variable. The function is guaranteed to return after release.

2.


The Affect of Thread Density on Run Time
Anderson
Lamport
time (s)
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
1
2
3
4
5
6
7
8
threads


The Affect of Thread Density on Run Time
Fischer
time (s)
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
threads

4a.


The Affect of Thread Density on Run Time
matrixMult
time (s)
0.070
0.065
0.060
0.055
0.050
0.045
0.040
1
2
3
4
5
6
7
8
threads