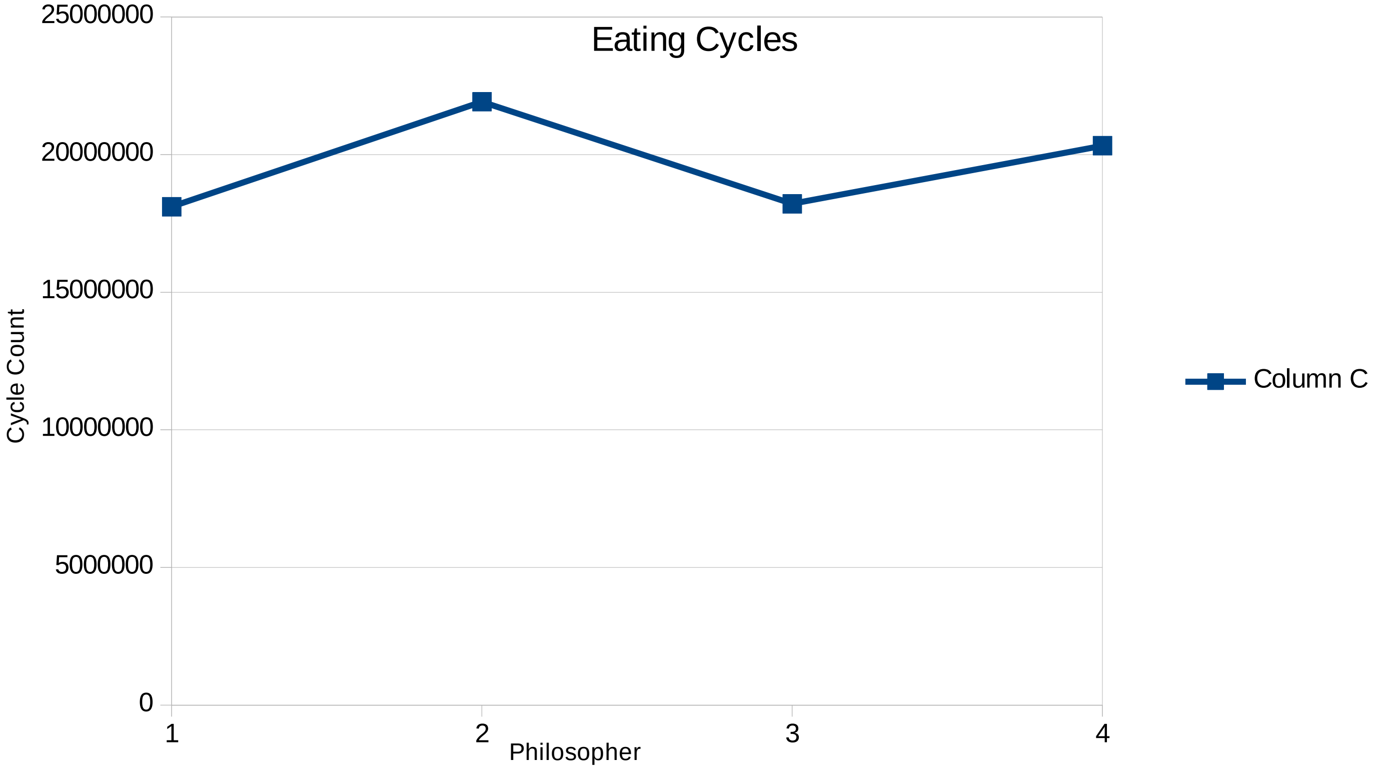
Patrick Hanrahan A53204304
Benjamin Hobbs A11317149
Assignment 4
17 August 2018

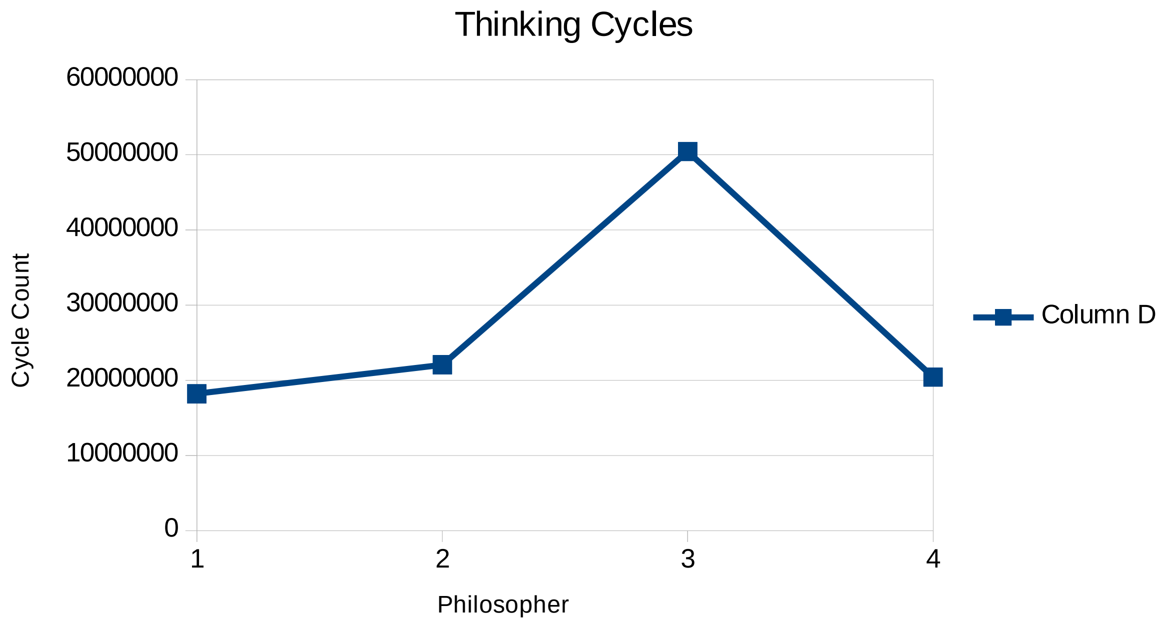
Part I: Dining Philosophers Problem

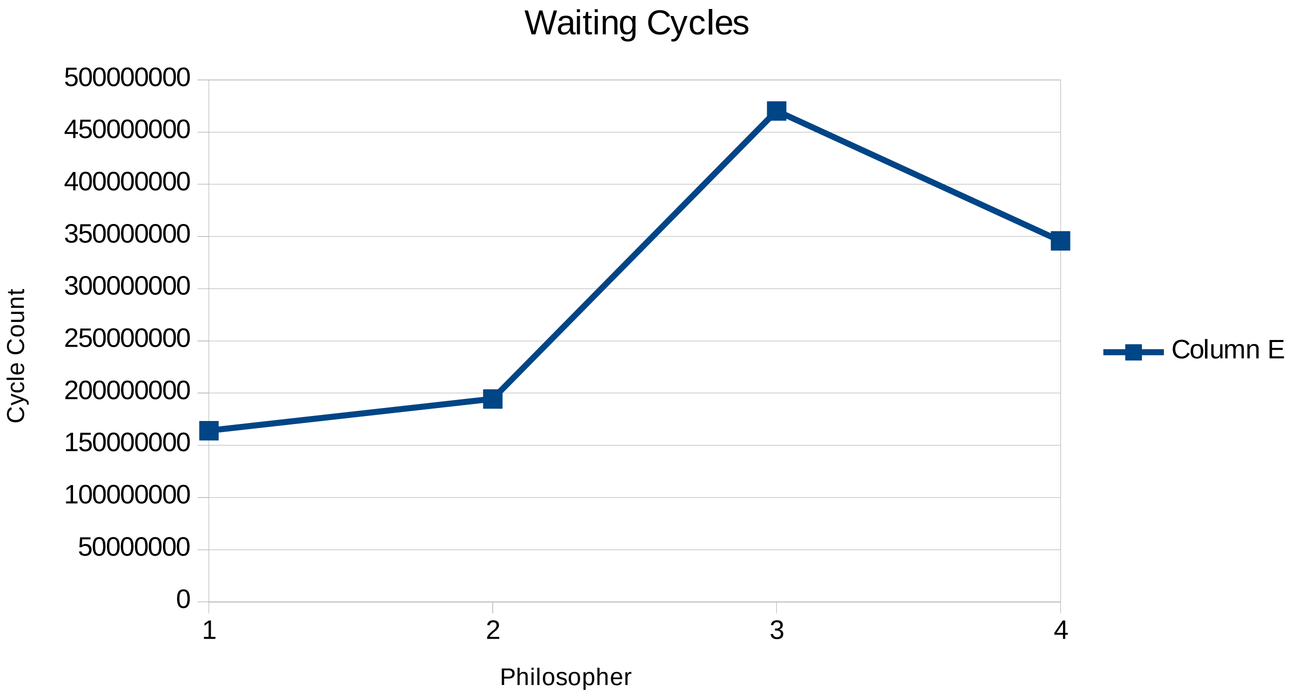
Please find our video clip at https://youtu.be/yMv7fmsL0YM

The following cycle time data was taken with the following command line execution:

```
./lab4 5 1000 10 100000
```







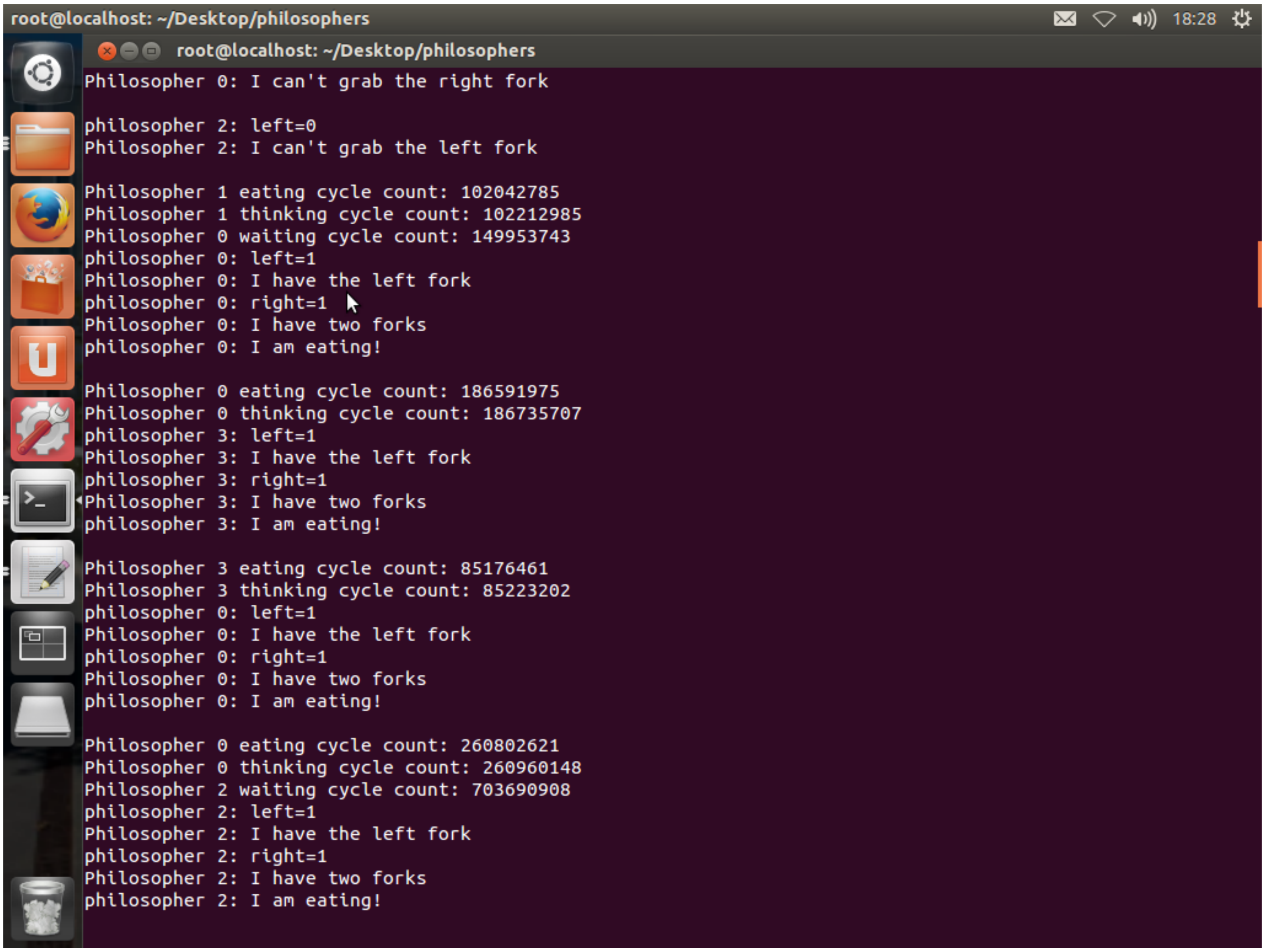
Part I summary:

We implemented a 4 element array of integers representing the four forks that we toggle between 1 (being available) and 0 (being unavailable). The solution was implemented in such a way that adjacent philosophers were never eating at the same time. The program accepts user command line arguments representing minimum and max eating time, as well as min and max thinking time as specified in the problem statement.

The meat of our program is in the “philosopher” function. We implement a pthread_mutex_t variable called “m” that allows access to a fork by controlling the left fork and right fork availability.

Once a philosopher is done eating, the forks are then made available to the other philosophers (the fork array element is toggled).

More detail is provided in the video.


root@localhost: ~/Desktop/philosophers
18:28
root@localhost: ~/Desktop/philosophers

```
Philosopher 0: I can't grab the right fork

philosopher 2: left=0
Philosopher 2: I can't grab the left fork

Philosopher 1 eating cycle count: 102042785
Philosopher 1 thinking cycle count: 102212985
Philosopher 0 waiting cycle count: 149953743
philosopher 0: left=1
Philosopher 0: I have the left fork
philosopher 0: right=1
Philosopher 0: I have two forks
philosopher 0: I am eating!

Philosopher 0 eating cycle count: 186591975
Philosopher 0 thinking cycle count: 186735707
philosopher 3: left=1
Philosopher 3: I have the left fork
philosopher 3: right=1
Philosopher 3: I have two forks
philosopher 3: I am eating!

Philosopher 3 eating cycle count: 85176461
Philosopher 3 thinking cycle count: 85223202
philosopher 0: left=1
Philosopher 0: I have the left fork
philosopher 0: right=1
Philosopher 0: I have two forks
philosopher 0: I am eating!

Philosopher 0 eating cycle count: 260802621
Philosopher 0 thinking cycle count: 260960148
Philosopher 2 waiting cycle count: 703690908
philosopher 2: left=1
Philosopher 2: I have the left fork
philosopher 2: right=1
Philosopher 2: I have two forks
philosopher 2: I am eating!
```

Part II: 2d Convolution

format: (q1,q2),float CPU cycles, fix point CPU cycles, RMSE,image quality of fix point.

(10,10), 286311, 231986, 0.00111181, image looks perfect for fix point

(10,10), 286774, 230557, 0.00111181, image looks perfect for fix point

(1,10), 250809, 230252, 0.80974600, image looks very discontinuous. discontinuities are spaced far apart.

(10,1), 286845, 230252, 0.28725800, image looks better than (1,10) but the discontinuities are more frequent.

(2,2), 250778, 230463, 0.38048400, image looks very similar to (10,1)

(3,3), 286362, 231252, 0.53009600, image looks bad.

(5,5), 286713, 230213, 0.03577590, image looks great but there exist small discontinuities.

Part II Summary:

As q1 and q2 get bigger, their RMSE generally decreases. This makes sense because as q1/q2 increase, their conversion from floating point to fixed point has less error/loss. As q1,q2 gets bigger, the CPU Cycles also gets bigger. This makes sense because its requiring more cycles to have more precision.Fixed point seems to use less CPU cycles, which is expected since fixed point is easier/faster. As RMSE decreases, of course the image looks better. This is because when RMSE is lower, you're seeing less precision loss. Also, (10,1) having an RMSE of 0.287 being lower than the RMSE of (1,10) 0.8 because you're using a float that is highly precise, doing all your calculations with that, and then converting. with (1,10) you're right off the bat using a low precision, so all your calculations will be very bad.