1. Drew Sadler
2. [asadler1@hopper3 Studio_13]$ gcc -o prime prime.c
   [asadler1@hopper3 Studio_13]$ ./prime
   Enter a number: 7
   Number is prime: 7
   [asadler1@hopper3 Studio_13]$ ./prime
   Enter a number: 23
   Number is prime: 23
   [asadler1@hopper3 Studio_13]$ ./prime
   Enter a number: 982451653
   Number is prime: 982451653
3. [asadler1@hopper3 Studio_13]$ ./prime
   Enter a number: 7
   Number is prime: 7
   Time elapsed is 1646692339 seconds
   (I don't think this is right)
4. [asadler1@hopper3 Studio_13]$ ./prime
   Number is prime: 2
   Number is prime: 3
   Number is prime: 5
   Number is prime: 7
   Number is prime: 11
   Number is prime: 13
   Number is prime: 17
   Number is prime: 19
   Number is prime: 23
   Number is prime: 29
   Number is prime: 31
   Number is prime: 37
   Number is prime: 41
   Number is prime: 43
   Number is prime: 47
   Number is prime: 53
   Number is prime: 59
   Number is prime: 61
   Number is prime: 67
   Number is prime: 71
   Number is prime: 73
   Number is prime: 79
   Number is prime: 83
   Number is prime: 89
   Number is prime: 97
5. It is logically independent because each number is a separate event from each other and does not interact with each other in any way, we know this also since we are having

to apply the algorithm each time for every new number, rather than building off of the last instance/number.

6. It's not independent of testing due to the fact that it's in a set and that since it's all in one event/action and it would just go through the set, as it would already know that the primes of that set would already work, and it's just about testing the different positions of the set against the number, technically building off of one another so that it's not entirely independent.
7. [asadler1@hopper3 Studio_13]$ ./parprime
   Number is prime: 71
   Number is prime: 73
   Number is prime: 2
   Number is prime: 3
   Number is prime: 5
   Number is prime: 97
   Number is prime: 31
   Number is prime: 83
   Number is prime: 23
   Number is prime: 11
   Number is prime: 13
   Number is prime: 67
   Number is prime: 89
   Number is prime: 7
   Number is prime: 79
   Number is prime: 61
   Number is prime: 59
   Number is prime: 29
   Number is prime: 41
   Number is prime: 43
   Number is prime: 17
   Number is prime: 19
   Number is prime: 37
   Number is prime: 53
   Number is prime: 47
   (^ No it does not match the output of the sequential version)
8. For (1000000)
   [asadler1@hopper3 Studio_13]$ ./parprime (parallel)
   Time elapsed is 1646708513 seconds
   [asadler1@hopper3 Studio_13]$ ./prime (sequential)
   Time elapsed is 1646708738 seconds

   For (20000000)
   [asadler1@hopper3 Studio_13]$ ./parprime (parallel)
   Time elapsed is 1646708917 seconds
   [asadler1@hopper3 Studio_13]$ ./prime (sequential)

Time elapsed is 1646709036 seconds
(I feel like this should show a larger time difference here, not making as much as a difference as probably expected)

9. In order to use pthreads in this instance we first would create separate threads to process for checking some doing the task of even numbers, and the others checking odds, so that it would be efficient on getting time done since both processes are running at once. As well as maybe splitting up the workload of the large 20000000 numbers, into even more threads so it would have to check less larger amounts and could have faster response time in smaller segments. We would also want to initialize and use the mutex lock and unlock each time when checking/editing the numbers in the for loop to check whether it is odd or even, so in the case that a thread would want to skip in front of the other it couldn't and be forced to wait and run in order. Finally joining the threads in the end after exhausting the for loop and giving the final result of both filled threads.