Instructions to Compile and run:

g++ ParallelPrimes.cpp

.\a.exe

Strategy:

The strategy for this Parallel prime finder is to create to data structures, a lockable counter to assign one prime to each thread and create a shared vector to ensure that no values are being over-written, and nothing can read as it is being written. The critical section is the portion of the sieve algorithm that updates all the shared primes for a number and controls the count of the number of primes which must be locked.

Experimental Analysis:

The experimental analysis was done by finding a verification source for the number of primes and comparing it to the output, the code was temporarily modified to take command line arguments which allowed it to modify the counted number of primes

Table 1. Values of π(*x*)

n	*x*	π(*x*)	ref
1	10	4	
2	100	25	
3	1,000	168	
4	10,000	1,229	
5	100,000	9,592	
6	1,000,000	78,498	
7	10,000,000	664,579	
8	100,000,000	5,761,455	
9	1,000,000,000	50,847,534	
10	10,000,000,000	455,052,511	

10^8

```
9.15821 5761455 279209790387276
99999787 99999821 99999827 99999839 99999847 99999931 99999941 99999959 99999971 99999989
```

100

```
PS C:\Users\benja\Desktop\Spring2024\Parallel\Assignment1> .\a.exe 100
0.0009995 25 1060
59 61 67 71 73 79 83 89 97 100
```

1,000

```
PS C:\Users\benja\Desktop\Spring2024\Parallel\Assignment1> .\a.exe 1000
0 168 76127
941 947 953 967 971 977 983 991 997 1000
```

100,000

```
PS C:\Users\benja\Desktop\Spring2024\Parallel\Assignment1> .\a.exe 100000
0.0090071 9592 454396537
99877 99881 99901 99907 99923 99929 99961 99971 99989 99991
```

As per the table above the values for the number of primes are accurate and due to the use of the efficient sieve algorithm and parallel processing the values of the sum are ensured to be correct as well