Mutex		Version 1		Version 2	
		computation time	Execution time	computation time	Execution time
p=4, n=1,600,000		0.023407s	0.030s	0.002285s	0.004s
p=1, n=1,600,000		0.009626s	0.018s	0.008241s	0.018s
p=5	n=8000	0.000231s	0.007s		
	n=1,600,000			0.004064s	0.006s

Discuss differences in the timing data for the two implementations. Do these results indicate a preference for either solution?
It doesn't seem that there is a big difference between version 1 and 2. However, Version 1 is slightly taking longer than version 2.

Are these solutions scalable? Explain.
Version 1, it is not strongly scalable because when n = 1600,000, the efficiency is decreasing as we add cores shown in table below.
Version 1, it is also not weakly scalable because when n and p increases in same ratio, the efficiency decreases.

Version 1	p=1	p=2	p=4	p=8	p=16
n=800000	0.00569	0.02756	0.01382	0.01052	0.01361
n=1600000	0.01114	0.03625	0.03230	0.02885	0.0268

Version 1 E	P=1	p=2	p=4	p=8	p=16
n=800000	1	0.10323	0.10293	0.06861	0.02613
n=1600000	1	0.154	0.086	0.048	0.0260

Version 2, is it not strongly scalable because the efficiency is decreasing when n is fixed size and adding cores.
Version 2, it is not weakly scalable because when n and p increases in same ratio, the efficiency decreases.

Version 2	p=1	p=2	p=4	p=8	p=16
n=800000	0.004934192	0.002693892	0.001408011	0.002226949	0.001885425
n=1600000	0.009649277	0.004823744	0.002638459	0.002664745	0.003071420

Version 2 E	P=1	p=2	p=4	p=8	p=16
n=800000	1	0.915811027	0.876092587	0.276959194	0.163563653
n=1600000	1	1.00018543	0.914290974	0.45263604	0.196352115