(20 marks) Compare the performance of the three implementations (in terms of time). In particular, increase the number of nodes that concurrently request to enter the critical sections. This is to investigate the performance trend for each of the protocol. For each experiment and for each protocol, compute the time between the first request and all the requesters exit the critical section. If you have ten nodes, therefore, your performance table should contain a total of 30 entries (ten entries for each of the three protocols).

	Concurrent nodes (in milliseconds)									
	1	2	3	4	5	6	7	8	9	10
Lamport Queue	126	250	380	493	622	704	867	1009	1125	1242
Ricart Argrawala	115	238	328	483	625	708	819	1011	1059	1213
Voting Protocol	127	241	344	499	635	-	-	-	-	-

The table above shows a sample of the runtime that was achieved on my machine. Results may differ from machine to machine. There is an expected 100ms wait for every critical section, therefore, the increment is expected to be at least 100ms per concurrent node added.

In general, Ricart Agrawala algorithm performs slightly better than Lamport Queue. This can be present by the less amount of processing required for message passing and maintaining the queue system. It is negligible in small amounts and the results may differ due to noise in the processing of the machine.

For Voting Protocol, the results are of average performance, almost on par with Lamport Queue and Ricart Agrawala, with very minute differences that is overshadowed by the expected wait and system noise.

However, for Voting Protocol, we are unable to progress to 6 or more concurrent nodes. This is as the nodes who are requesting are not allowed to vote for someone else other than themselves, therefore causing a situation where nobody wins. It can be further justified by proving that for 6 concurrent requests, any requesting node could only have a maximum of 5 votes, which does not satisfy the win condition.