Nishanth Muruganandam 110276247

The three algorithms chosen for this assignment are:

1. Lamport Mutual Exclusion Algorithm
2. Ricart & Agrawala Time-stamp based mutual exclusion algorithm
3. Ricart & Agrawala Token based mutual exclusion algorithm

Tool used:

The tool that has been used is TLA+. It is integrated into Eclipse-IDE and TLC model checker ships with it. It is highly sophisticated and writing a TLA specification requires the user to know only how to write algorithm in Plus-Cal and the tool does the translation to the TLA specification.

Credits:

Plus-Cal version of the algorithms are taken from Lamport Handout and previous batch students' projects. However, there were many bugs in the previous projects, so I ended up modifying most of the source code. I ran the model checker on these specification to find these errors. Some failed all the correctness properties. I used Error-Trace of the tool to single out the erroneous step and happened to find that the implementation itself was flawed in many places.

Limitation on the TLA specification to check distributed algorithms:

Unlike DistAlgo, we don't have an abstraction over the communication overhead between the processes/sites. So, the user has to write a layer of communication network through which the sites should communicate. TLA doesn't ship this as a in-built Module. (*I personally would like to comment that including such a communication module in the standard modules would reduce the overhead on the user and avoid redundant codes.)*

Comparison based on **SPECIFICATION** of the algorithms:

Attributes	**Lamport Mutual Exclusion**	**Ricart & Agrawala Timetamp**	**Ricart & Agrawala Token based**
Size(PlusCal)	~105 lines	~135 lines	~138 lines
Size (TLA spec)	~115 lines	~200 lines	~200 lines
Ease of understanding	Very Easy	Easy	Easy
Safety	Yes	Yes	Yes
Liveness	Yes	Yes	Yes
Fairness	Yes	Yes	Yes

How was the checking done?

A variable called **STATE**(*Can be found the .tla file)* is added to each process. This state is updated to 'idle','waiting' & 'owner' whichever is appropriate at that step.

1. To check for safety, an invariant is added and in the TLC model checker it is specified as an invariant so that it will be checked if its true in every reachable state.

2. For liveness and fairness, they have been added as a temporal formulae in the model checker.

(Formulae for all these three can be found in the end in .tla files)
Note: Liveness in TLA is expressed with Fairness. [taken from Lamport Slides]

Comparison based on **Checkers/Provers** of the algorithms:

The time taken to set up checkers/provers for all the algorithms was low only. Majority of the time goes in writing the specification of the algorithm overcoming some of the limitation as mentioned before. Please find below the summary on the result of running checkers/provers on the algorithms:

(Note: The number of worker threads count is set to 5. The default value is 2)

-Lamport Mutual Exclusion algorithm:

No of Processes	**MaxClock**	**Elapsed Time**	**States found**	**Distinct state**	**Errors Detected**
2	3	<10 seconds	164543	52509	No
3	3	~40 seconds	582844	101086	No
3	5	>15 min	>745437	>156564	No
4	3	>30 min	>7934731	>1515713	No

(Note: '>x' is the time till it was allowed to run and then killed the execution as my laptop started to crash.)

-Ricart & Agrawala Mutual Exclusion algorithm – Timestamp based:

No of Processes	**MaxClock**	**Elapsed Time**	**States found**	**Distinct state**	**Error Detected**
2	2	5 seconds	150029	150029	No
2	3	~80 seconds	3543008	996987	No
3	3	>30 min	>5027763	>1761401	No
3	5	>1 hour	>9567751	>2452134	No

(Note: '>x' is the time till it was allowed to run and then killed the execution as my laptop started to crash.)

-Ricart & Agrawala Mutual Exclusion algorithm – Token Based:

No of Processes	**MaxClock**	**Elapsed Time**	**States found**	**Distinct state**	**Error Detected**
2	2	2 seconds	10561	2880	No
2	3	3 seconds	10561	2880	No
3	3	17 seconds	857328	164763	No
3	5	30 seconds	927567	167496	No

Max Clock is a constant value up to which the clock will increment. Basically, it acts as a termination condition.

Possibility of extra credit:

I found a bug in older version of the TLA+ toolbox. It was a concurrency issue. The thread races to check a value before the variable has been actually created. This bug was fixed in the later version of the toolbox. So, I had informed the professor about this so that other students might not get into the trouble that I got into.

This bug was brought to light when I ran my safety model checker on the Lamport Mutual Exclusion algorithm. Also, I fixed most of the bugs in the previous batch projects.