ASSIGNMENT 3
NAME: VIVEK BANSAL
STUDENT ID: 112044493
SUBJECT: ASYNCHRONOUS SYSTEMS (CSE 535)
DUE DATE: 07 OCTOBER 2018

Ans 1:

Spec1: Lamport
Spec2: Lamport PODC 2000
Spec3: Merz

	Spec1	Spec2	Spec3
Specification Size (excluding blank lines and comments)	90	77	113
Ease of understanding	Easy (code is properly written and commented so easy to understand).	Medium (no comments in the code and have to figure out how to give constants values, have to change Nat value in the code for each different run).	Hard (code is using IF-ELSE many times, nesting is at deeper level which made it difficult to analyze the complete flow of code).
How closely are different aspects of algorithms followed	When a process needs to access CS, it broadcasts request messages to all other processes. Other processes send ack to this process, and this process executes in CS. After this it sends release message to other processes to notify them about release of CS.	When a process needs to access CS, it goes from “idle” to “waiting” state and sends “acquire” command to other processes. Other processes send back ack after which current process becomes “owner” of CS. After executing CS, current process sends “release” msg to other processes and move from “owner” to “idle” state.	Each node is comprised of 2 processes: “site” process that requests access to CS, and “communicator” which receives messages directed to the site. So we have n sites and n communicators. Communicator is acting as a message queue for the site process. When process exits from CS, it sends release notification to other processes.

Properties	Safety Violated: NO Liveness Violated: YES Spec is violating liveness as the system can stutter and no progress will be made in that behavior. So weak fairness is added to the spec as it ensures that system will progress. Mutex == \A p,q \in crit : p = q Live == \A p \in Proc: <>(p \in crit) Spec == Init /\ [][Next]_vars WF(A) == <>[](ENABLED A) => []<>(A) SF(A) == []<>(ENABLED A) => []<>(A) NewSpec == Init /\ [][Next]_vars /\ WF_vars(Next)	Safety Violated: NO Liveness Violated: NO Spec is following safety and liveliness. Mutex == \A p,q \in Proc:state[p] = "owner" /\ state[q] = "owner" => p = q Liveness == \A p \in Proc : /\ WF_vars(Acquire(p)) /\ \A q \in Proc \ {p} : WF_vars(RcvMsg(p, q)) Spec == Init /\ [][Next]_vars /\ Liveness	Safety Violated: NO Liveness Violated: YES Spec is violating liveness due to stuttering so fairness is added to the spec to ensure progress of the system. Mutex == \A s,t \in Sites : pc[s] = "crit" /\ pc[t] = "crit" => s = t Liveness == \A s \in Sites : pc[s] = "enter" ~> pc[s] = "crit" Spec == Init /\ [][Next]_vars Fairness == /\ \A s \in Sites: WF_vars(enter(s)) /\ WF_vars(crit(s)) /\ WF_vars(exit(s)) /\ \A c \in Comms: WF_vars(comm(c)) LiveSpec == Spec /\ Fairness

Ans2:

Spec1: Lamport
Spec2: Lamport PODC 2000
Spec3: Merz

Below table is with mutex enabled:

Procs	Nats	States (Spec1)	States (Spec2)	States (Spec3)	Time (Spec1)	Time (Spec2)	Time (Spec3)	Safety Violated Spec1	Spec2	Spec3
2	5	1.3K	16K	23K	00:00:03	00:00:03	00:00:04	NO	NO	NO
2	6	2.0K	59K	43K	00:00:03	00:00:04	00:00:04	NO	NO	NO
2	7	2.8K	186K	67K	00:00:02	00:00:04	00:00:04	NO	NO	NO
2	8	3.7K	513K	95K	00:00:02	00:00:04	00:00:04	NO	NO	NO
2	9	4.8K	1285K	128K	00:00:03	00:00:04	00:00:04	NO	NO	NO
3	2	2.2K	1K	21K	00:00:03	00:00:06	00:00:04	NO	NO	NO
3	3	41K	75K	578K	00:00:03	00:00:06	00:00:07	NO	NO	NO
3	4	276K	1978K	7763K	00:00:04	00:00:21	00:00:50	NO	NO	NO
3	5	1033K	28345K	48063K	00:00:08	00:02:47	00:05:02	NO	NO	NO
3	6	2729K	130000K	187791K	00:00:12	00:07:05	00:19:33	NO	NO	NO
4	2	68K	1089K	871K	00:00:06	00:00:05	00:01:08	NO	NO	NO
4	3	7302K	47405K	8712K	00:00:25	00:03:13	00:20:09	NO	NO	NO

Below table is with liveness enabled:
(Note: I am checking with Spec without fairness incorporated into it)

Procs	Nats	States (Spec1)	States (Spec2)	States (Spec3)	Time (Spec1)	Time (Spec2)	Time (Spec3)	Liveness Violated Spec1	Spec2	Spec3
2	5	1.3K	16K	23K	00:00:03	00:00:03	00:00:03	YES	NO	YES
2	6	2.0K	59K	43K	00:00:02	00:00:04	00:00:02	YES	NO	YES
2	7	2.8K	186K	67K	00:00:02	00:00:09	00:00:02	YES	NO	YES
2	8	3.7K	513K	95K	00:00:02	00:00:18	00:00:03	YES	NO	YES
2	9	4.8K	1285K	128K	00:00:01	00:00:39	00:00:03	YES	NO	YES
3	2	2.2K	1K	21K	00:00:01	00:01:40	00:00:02	YES	NO	YES
3	3	41K	75K	178K	00:00:03	00:02:20	00:00:05	YES	NO	YES
3	4	276K	1978K	145K	00:00:05	00:03:26	00:00:04	YES	NO	YES
3	5	306K	28345K	124K	00:00:04	00:07:50	00:00:04	YES	NO	YES
3	6	253K	130000K	144K	00:00:04	00:21:00	00:00:04	YES	NO	YES
4	2	68K	1089K	165K	00:00:02	00:00:21	00:00:04	YES	NO	YES
4	3	233K	47405K	129K	00:00:04	00:50:02	00:00:04	YES	NO	YES

Below table is when we have added fairness to Spec1 and Spec3:

Procs	Nats	States (Spec1)	States (Spec3)	Time (Spec1)	Time (Spec3)	Liveness Violated	
						Spec1	Spec3
2	5	1.3K	23K	00:00:03	00:00:03	NO	NO
2	6	2.0K	43K	00:00:02	00:00:02	NO	NO
2	7	2.8K	67K	00:00:02	00:00:02	NO	NO
2	8	3.7K	95K	00:00:02	00:00:03	NO	NO
2	9	4.8K	128K	00:00:03	00:00:03	NO	NO
3	2	2.2K	21K	00:00:03	00:00:02	NO	NO
3	3	41K	578K	00:00:03	00:00:06	NO	NO
3	4	276K	7763K	00:00:05	00:00:50	NO	NO
3	5	1033K	48063K	00:00:08	00:05:02	NO	NO
3	6	2729K	187791K	00:00:12	00:19:33	NO	NO
4	2	68K	871K	00:00:06	00:01:08	NO	NO
4	3	7302K	8712K	00:00:25	00:20:09	NO	NO

Ease of setting up and running:
a) Spec1 was easy to run as it was taking less time for different number of processes and clock values as compared to other specs.
b) For spec2, it took some time to figure out like how to give constants values to Proc and \ll(p,q). Also, I have to specify Nat value for each run in the code itself to set the maximum value of clock, rather than giving value in the constants. So I added new constant maxClock to specify value in the model itself rather that in the code.
c) Spec3 was easy to setup and run but it was taking too much time to explore the states in case of mutex enabled.

Explanations of experiments performed:

1) Spec1 (Lamport): tla file is LamportMutex.tla. I added value of constants N and maxClock in the model to check for correctness criteria. I gave mutex condition in the invariant and liveness in the properties to run the model.
2) Spec2 (Lamport PODC 2000): tla file is LamportOwn2000.tla. I provided constants value of proc value as set (like {1,2,3}) and conditions specified on processes as \ll(p,q) <- (p < q). First, I ran the model to check for mutex invariant and then I checked for liveness property by specifying in the model.
3) Spec3 (Merz): tla file is MutexLamport.tla. I added constants value of maxClock and N in the model and also specified the Nat value in the code to set the maximum value of the clock. Then I checked for mutex and liveness separately by running the model for different value of processes and clock values.

References:

1) Spec1 taken from https://github.com/tlaplus/Examples/blob/master/specifications/lamport_mutex/LamportMutex.tla
2) Spec 2 is taken from http://www.cs.stonybrook.edu/~liu/distalgo/lamutex_tlaplus_lamport_handout.pdf
3) Spec 3 is taken from http://www.cs.stonybrook.edu/~liu/distalgo/lamutex_pluscal_merz

----------END----------