COMPSYS - 705 Nkmu576
Assignment 2 286933069
Q1)
Safety Property:

```
//LTL: Safety property-1
ltl safety {[](mutex <= 1)}
```

The first property of safety which is that multiple processes cannot enter the critical section was checked using this LTL formula. It is checking the bounds of the mutex, such that it does not exceed 1, which if it did, would mean that multiple processes have obtained the mutex, meaning multiple processes have entered the critical section.

```
//LTL: Liveness property-2, checking if for any process, once it is waiting, it will enter the critical state
//Indicated by the position array has a 1, meaning waiting, and then once set to zero, meaning process finished
ltl Liveness01 {[] (((pos[0] == 1) -> <>(pos[0] == 0) -> <>(CriticalArray[0]))
                ||   ((pos[1] == 1) -> <>(pos[1] == 0) -> <>(CriticalArray[1]))
                ||   ((pos[2] == 1) -> <>(pos[2] == 0) -> <>(CriticalArray[2])) )}

//LTL: Liveness property-2, checking if for ALL process, once it is waiting, it will enter the critical state
//Indicated by the position array has a 1, meaning waiting, and then once set to zero, meaning process finished
ltl Liveness02 {[] (((pos[0]) -> <>(pos[0] == 0) -> <>(CriticalArray[0]))
                &&   ((pos[1]) -> <>(pos[1] == 0) -> <>(CriticalArray[1]))
                &&   ((pos[2]) -> <>(pos[2] == 0) -> <>(CriticalArray[2])) )}
```

The second property of liveness is that if a process is waiting, it will eventually enter the critical section. This is checked using two LTL formulas, to verify that it happens to any process and to all processes. The basic line contained inside the LTL formula is
`(pos[0] == 1) -> <>(pos[0] == 0) -> <>(CriticalArray[0])`
This formula checks the position array for the process, and checks if it is equal to a 1, meaning that the process is now in the waiting state, the formula then makes a logical implication that eventually the position array will be set to zero, which implies that the process has finished waiting, entered the critical section and has since completed operation. This action itself also implies that the CriticalArray, accessed in the critical section, is incremented, further ensuring that the critical section has been entered.

In the Liveness01 LTL, these three lines for the three processes, have logical ors in between each, to check if the operation is completed successfully on any process, always. The Liveness02 has logical ands in between each line, to check if this action occurs for all processes, always.

```
//LTL: Liveness property-3, Any process not in the critical section will eventually enter the critical section
ltl Liveness03 {[]((pos[0] == 1) -> <>(CriticalArray[0] == 1)
                || (pos[1] == 1) -> <>(CriticalArray[1] == 1)
                || (pos[2] == 1) -> <>(CriticalArray[2] == 1) )}

//LTL: Liveness property-3, Any process not in the critical section will eventually enter the critical section
ltl Liveness04 {[]((CriticalArray[0] == 0) -> <>(CriticalArray[0] == 1)
                && (CriticalArray[1] == 0) -> <>(CriticalArray[1] == 1)
                && (CriticalArray[2] == 0) -> <>(CriticalArray[2] == 1) )}
```

The last two LTL functions check the property of liveness, where any process not in the critical section will eventually enter the critical section.

In the Liveness03 function, this is checked by checking if the process has entered the waiting state, meaning that they have yet to enter the critical section. If they are yet to enter the critical section, the LTL logically implies that eventually, the CriticalArray will be set to one, meaning that the process has indeed entered the critical section, to increment the array.

The Liveness04 function checks the property by checking if the critical section array has been set to zero, then logically implies that eventually the critical section array will be incremented by 1, meaning that the process has entered the critical section.

Outputs:
Safety - Property 1

```
(base) nikhil@pop-os:~/Documents/COMPSYS/705Avinish/code/Assignment02/promela$ ./pan -a -N safety
pan: ltl formula safety

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (safety)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 56 byte, depth reached 211, errors: 0
     4623 states, stored
     1477 states, matched
     6100 transitions (= stored+matched)
        2 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.370       equivalent memory usage for states (stored*(State-vector + overhead))
    0.455       actual memory usage for states
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  128.925       total actual memory usage


unreached in proctype P
        (0 of 32 states)
unreached in init
        (0 of 5 states)
unreached in claim safety
        _spin_nvr.tmp:8, state 10, "-end-"
        (1 of 10 states)

pan: elapsed time 0 seconds
```

Liveness01 - Property 2

```
(base) nikhil@pop-os:~/Documents/COMPSYS/705Avinish/code/Assignment02/promela$ ./pan -a -N Liveness01
pan: ltl formula Liveness01

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (Liveness01)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 56 byte, depth reached 211, errors: 0
    28407 states, stored (31380 visited)
    39182 states, matched
    70562 transitions (= visited+matched)
       18 atomic steps
hash conflicts:        24 (resolved)

Stats on memory usage (in Megabytes):
    2.276       equivalent memory usage for states (stored*(State-vector + overhead))
    1.920       actual memory usage for states (compression: 84.36%)
                state-vector as stored = 43 byte + 28 byte overhead
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  130.390       total actual memory usage


unreached in proctype P
        (0 of 32 states)
unreached in init
        (0 of 5 states)
unreached in claim Liveness01
        _spin_nvr.tmp:73, state 100, "-end-"
        (1 of 100 states)

pan: elapsed time 0.02 seconds
pan: rate   1569000 states/second
```

Liveness02 - Property 2

```
(base) nikhil@pop-os:~/Documents/COMPSYS/705Avinish/code/Assignment02/promela$ ./pan -a -N Liveness02
pan: ltl formula Liveness02

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (Liveness02)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 56 byte, depth reached 211, errors: 0
    28479 states, stored (40407 visited)
    48716 states, matched
    89123 transitions (= visited+matched)
       14 atomic steps
hash conflicts:        44 (resolved)

Stats on memory usage (in Megabytes):
    2.281       equivalent memory usage for states (stored*(State-vector + overhead))
    1.920       actual memory usage for states (compression: 84.16%)
                state-vector as stored = 43 byte + 28 byte overhead
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  130.390       total actual memory usage


unreached in proctype P
        (0 of 32 states)
unreached in init
        (0 of 5 states)
unreached in claim Liveness02
        _spin_nvr.tmp:112, state 54, "-end-"
        (1 of 54 states)

pan: elapsed time 0.02 seconds
pan: rate   2020350 states/second
```

Liveness03 - Property 3

```
(base) nikhil@pop-os:~/Documents/COMPSYS/705Avinish/code/Assignment02/promela$ ./pan -a -N Liveness03
pan: ltl formula Liveness03

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (Liveness03)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 56 byte, depth reached 211, errors: 0
    14261 states, stored (19943 visited)
    21670 states, matched
    41613 transitions (= visited+matched)
        4 atomic steps
hash conflicts:        13 (resolved)

Stats on memory usage (in Megabytes):
    1.142       equivalent memory usage for states (stored*(State-vector + overhead))
    1.041       actual memory usage for states (compression: 91.11%)
                state-vector as stored = 49 byte + 28 byte overhead
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  129.511       total actual memory usage


unreached in proctype P
        (0 of 32 states)
unreached in init
        (0 of 5 states)
unreached in claim Liveness03
        _spin_nvr.tmp:134, state 29, "-end-"
        (1 of 29 states)

pan: elapsed time 0.02 seconds
pan: rate    997150 states/second
```

Liveness04 - Property 3

```
(base) nikhil@pop-os:~/Documents/COMPSYS/705Avinish/code/Assignment02/promela$ ./pan -a -N Liveness04
pan: ltl formula Liveness04

(Spin Version 6.5.1 -- 20 December 2019)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (Liveness04)
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 56 byte, depth reached 211, errors: 0
    12560 states, stored (16541 visited)
    20231 states, matched
    36772 transitions (= visited+matched)
        6 atomic steps
hash conflicts:         8 (resolved)

Stats on memory usage (in Megabytes):
    1.006       equivalent memory usage for states (stored*(State-vector + overhead))
    0.943       actual memory usage for states (compression: 93.74%)
                state-vector as stored = 51 byte + 28 byte overhead
  128.000       memory used for hash table (-w24)
    0.534       memory used for DFS stack (-m10000)
  129.413       total actual memory usage


unreached in proctype P
        (0 of 32 states)
unreached in init
        (0 of 5 states)
unreached in claim Liveness04
        _spin_nvr.tmp:156, state 29, "-end-"
        (1 of 29 states)

pan: elapsed time 0 seconds
```

All outputs show successful completion of the model checking using LTL. As all the outputs from running the Pan file, checking all the against the various LTL formulas, does not return any errors, or any acceptance loops detected, we can safely say that the implementation in Promela does indeed satisfy the LTL formulas.

Q2)
The output from the Z3 Solver is

```
(base) nikhil@pop-os:~/Documents/COMPSYS/705Avinish/code/Assignment02$ python3 ./SMT.py
Circuits are equivalent
(base) nikhil@pop-os:~/Documents/COMPSYS/705Avinish/code/Assignment02$
```

The circuits are equivalent.
To check this, the implementation and specification were both implemented using Z3. The resulting outputs from the two functions are then inputted into a XOR function, which will check any differences between the two. If the XOR returns true, this means that the that the implementations are different, as the function of an exclusive can be found by the truth table:

Input		**Output**
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

As such, the outputs of the implementation and spec are both the same, thus, the xor outputs a zero.