

ASYNCHRONOUS ASSIGNMENT

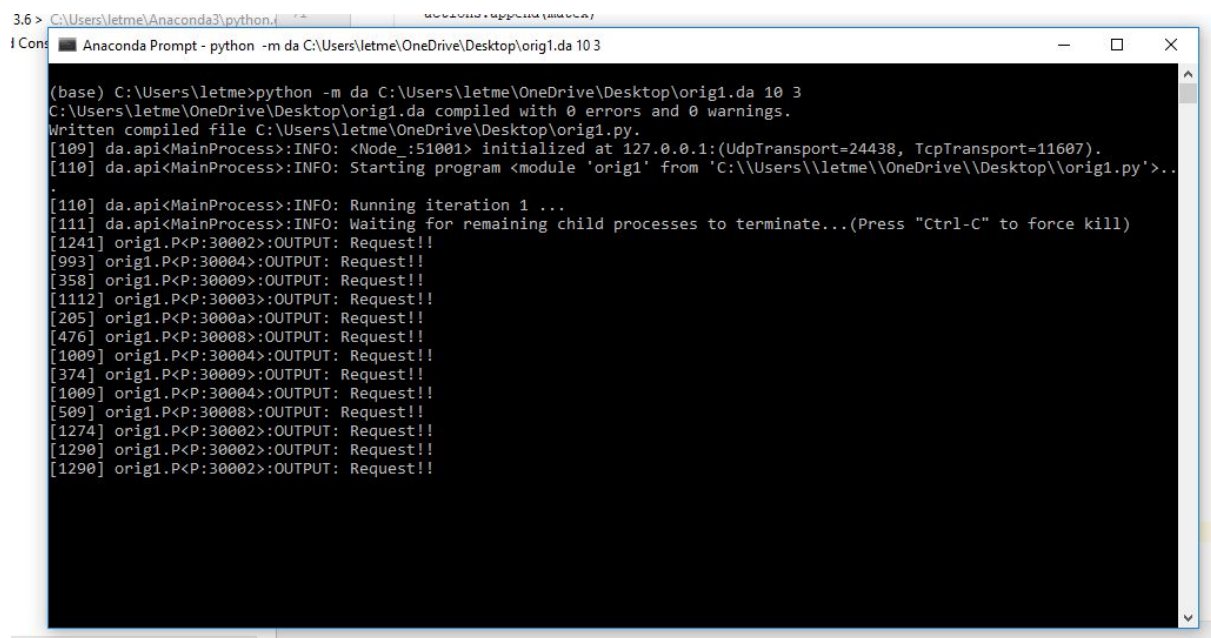
Question 2 :

Describe at least 2 precise ways that the algorithm does not satisfy the specified safety and liveness requirements.

Answer 2:

Liveness violation : The liveness will be violated if not even one process goes into the critical section. The liveness violation can be found in lamport's mutex algorithm if:

1. There is a deadlock in the processes when the processes are asking for critical section, so even before they get the critical section, the system will go into deadlock. This can happen when, we have requests which are all



```
3.6> C:\Users\letme\Anaconda3\python...
I Cons Anaconda Prompt - python -m da C:\Users\letme\OneDrive\Desktop\orig1.da 10 3

(base) C:\Users\letme>python -m da C:\Users\letme\OneDrive\Desktop\orig1.da 10 3
C:\Users\letme\OneDrive\Desktop\orig1.da compiled with 0 errors and 0 warnings.
Written compiled file C:\Users\letme\OneDrive\Desktop\orig1.py.
[109] da.api<MainProcess>:INFO: <Node :51001> initialized at 127.0.0.1:(UdpTransport=24438, TcpTransport=11607).
[110] da.api<MainProcess>:INFO: Starting program <module 'orig1' from 'C:\\Users\\letme\\OneDrive\\Desktop\\orig1.py'>..
[110] da.api<MainProcess>:INFO: Running iteration 1 ...
[111] da.api<MainProcess>:INFO: Waiting for remaining child processes to terminate...(Press "Ctrl-C" to force kill)
[1241] orig1.P<P:30002>:OUTPUT: Request!!
[993] orig1.P<P:30004>:OUTPUT: Request!!
[358] orig1.P<P:30009>:OUTPUT: Request!!
[1112] orig1.P<P:30003>:OUTPUT: Request!!
[205] orig1.P<P:3000a>:OUTPUT: Request!!
[476] orig1.P<P:30008>:OUTPUT: Request!!
[1009] orig1.P<P:30004>:OUTPUT: Request!!
[374] orig1.P<P:30009>:OUTPUT: Request!!
[1009] orig1.P<P:30004>:OUTPUT: Request!!
[509] orig1.P<P:30008>:OUTPUT: Request!!
[1274] orig1.P<P:30002>:OUTPUT: Request!!
[1290] orig1.P<P:30002>:OUTPUT: Request!!
[1290] orig1.P<P:30002>:OUTPUT: Request!!
```

As we can see here, none of the processes went into critical section. All of the processes are still in request mode. In this situation the deadlock was created. The problem here was the random calling of request, critical_section, release which was interleaved.

Safety Violation :

Safety violation happens when there are more than 2 requests in the critical section. The safety violation can cause deadlock. As the processes are random, we will have a request, critical_section and release interleaved, we will have the requests of 1 process getting generated for critical_section and getting served, and also, another process which also gets its request served in critical section.

As both the processes are in critical section we will have safety violated as for safety just one out of all processes should be in cs at a single time.

```
(base) C:\Users\letme>python -m da C:\Users\letme\OneDrive\Desktop\orig1.da 9 7
[94] da.api<MainProcess>:INFO: <Node:dd001> initialized at 127.0.0.1:(UdpTransport=24315, TcpTransport=16696).
[94] da.api<MainProcess>:INFO: Starting program <module 'orig1' from 'C:\\Users\\letme\\OneDrive\\Desktop\\orig1.py'>...
[95] da.api<MainProcess>:INFO: Running iteration 1 ...
[120] da.api<MainProcess>:INFO: Waiting for remaining child processes to terminate...(Press "Ctrl-C" to force kill)
[905] orig1.P<P:e2008>:OUTPUT: Request!!
[2849] orig1.P<P:e2003>:OUTPUT: Request!!
[2862] orig1.P<P:e2003>:OUTPUT: Request!!
[2862] orig1.P<P:e2003>:OUTPUT: Request!!
[2867] orig1.P<P:e2003>:OUTPUT: Request!!
[2873] orig1.P<P:e2003>:OUTPUT: in cs
[2874] orig1.P<P:e2003>:OUTPUT: Request!!
[2875] orig1.P<P:e2003>:OUTPUT: Request!!
[2875] orig1.P<P:e2003>:OUTPUT: Request!!
[2880] orig1.P<P:e2003>:OUTPUT: in cs
[2880] orig1.P<P:e2003>:OUTPUT: in cs
[2880] orig1.P<P:e2003>:OUTPUT: Request!!
```

As we can see here, that for multiple processes with multiple requests, we are getting the deadlock on the condition as, none of the requests released the critical section, but there would be multiple requests for critical section and maybe more than one request got into critical_section. Therefore, we have a deadlock at this situation and thus the safety is violated.

Reference - for question-1, I discussed the solution with TA-christopher and took his help. Also I discussed with one of my peers- Ishika agarwal (111971057).