In the code we start process counter (cnt) at 0 but finish at 4, for a total of 5 blocks. In the timing diagram, we have 4 blocks. There appears to be an error in the timing diagram for priority inversion.

## 2.1 Problem Inversion Scenario - Priority Inheritance Output

tick=0 scheduler runs , active\_p=3

P3->[0]

tick=1 scheduler runs , active\_p=3

P3->[1]

.....Attempting to Lock Semaphore by ..

..... Semaphore locked by P3

tick=2 scheduler runs , active\_p=2

P2->[0]

tick=3 scheduler runs , active\_p=2

P2->[1]

tick=4 scheduler runs , active\_p=1

P1->[0]

tick=5 scheduler runs , active\_p=1

P1->[1]

.....Attempting to Lock Semaphore ..

......P1 blocked ..

tick=6 scheduler runs , active\_p=3

P3->[2]

tick=7 scheduler runs , active\_p=3

P3->[3]

.....Unlocking Semaphore by ..

......semaphore unlocked by P3 ..

......P1 unblocked ..

..... Semaphore locked by P1

tick=8 scheduler runs , active\_p=1

P1->[2]

tick=9 scheduler runs , active\_p=1

P1->[3]

.....Unlocking Semaphore ..

......semaphore unlocked by P1 ..

tick=10 scheduler runs , active\_p=1

P1->[4]

.........P1 thread ends.........

tick=11 scheduler runs , active\_p=2

P2->[2]

tick=12 scheduler runs , active\_p=2

P2->[3]

tick=13 scheduler runs , active\_p=2

P2->[4]

tick=14 scheduler runs , active\_p=2

P2->[5]

tick=15 scheduler runs , active\_p=2

P2->[6]

.........P2 thread ends.........

tick=16 scheduler runs , active\_p=3

P3->[4]

tick=17 scheduler runs , active\_p=3

P3->[5]

.........P3 thread ends.........

tick=18 scheduler runs

tick=19 scheduler runs

tick=20 scheduler runs

tick=21 scheduler runs

tick=22 scheduler runs

tick=23 scheduler runs

tick=24 scheduler runs

tick=25 scheduler runs

tick=26 scheduler runs

tick=27 scheduler runs

tick=28 scheduler runs

tick=29 scheduler runs

## 2.2 Problem Inversion Scenario - Priority Ceiling Output

tick=0 scheduler runs , active\_p=3

P3->[0]

tick=1 scheduler runs , active\_p=3

P3->[1]

.....Attempting to Lock Semaphore by ..

..... Semaphore locked by P3

tick=2 scheduler runs , active\_p=2

P2->[0]

tick=3 scheduler runs , active\_p=2

P2->[1]

tick=4 scheduler runs , active\_p=1

P1->[0]

tick=5 scheduler runs , active\_p=1

P1->[1]

.....Attempting to Lock Semaphore ..

......P1 blocked ..

tick=6 scheduler runs , active\_p=3

P3->[2]

tick=7 scheduler runs , active\_p=3

P3->[3]

.....Unlocking Semaphore by ..

......semaphore unlocked by P3 ..

......P1 unblocked ..

..... Semaphore locked by P1

tick=8 scheduler runs , active\_p=1

P1->[2]

tick=9 scheduler runs , active\_p=1

P1->[3]

.....Unlocking Semaphore ..

......semaphore unlocked by P1 ..

tick=10 scheduler runs , active\_p=1

P1->[4]

.........P1 thread ends.........

tick=11 scheduler runs , active\_p=2

P2->[2]

tick=12 scheduler runs , active\_p=2

P2->[3]

tick=13 scheduler runs , active\_p=2

P2->[4]

tick=14 scheduler runs , active\_p=2

P2->[5]

tick=15 scheduler runs , active\_p=2

P2->[6]

.........P2 thread ends.........

tick=16 scheduler runs , active\_p=3

P3->[4]

tick=17 scheduler runs , active\_p=3

P3->[5]

.........P3 thread ends.........

tick=18 scheduler runs

tick=19 scheduler runs

tick=20 scheduler runs

tick=21 scheduler runs

tick=22 scheduler runs

tick=23 scheduler runs

tick=24 scheduler runs

tick=25 scheduler runs

tick=26 scheduler runs

tick=27 scheduler runs

tick=28 scheduler runs

tick=29 scheduler runs

## 2.3 Deadlock Scenario - Priority Inheritance Output

tick=0-> scheduler runs (P1=0; P2=0.5), active\_p = P2

P2->[0]

tick=1-> scheduler runs (P1=0; P2=0.5), active\_p = P2

P2->[1]

..... P2-CS2-entry ..

..... semaphore 2 locked by P2

tick=2-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[1]

..... P1-CS1-entry ..

..... semaphore 1 locked by P1

tick=3-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[2]

..... P1-CS2-entry ..

......P1 attempted to lock semaphore 2, already locked by P2 ..

......P1 blocked ..

tick=4-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

P2->[2]

..... P2-CS1-entry ..

......P2 attempted to lock semaphore 1, already locked by P1 ..

......P2 blocked ..

tick=5-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=6-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=7-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=8-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=9-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=10-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=11-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=12-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=13-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=14-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=15-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=16-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=17-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=18-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=19-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=20-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=21-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=22-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=23-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=24-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=25-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=26-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=27-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=28-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

tick=29-> scheduler runs (P1=0.7; P2=0.7), active\_p = P2

## 2.4 Deadlock Scenario - Priority Ceiling Output

tick=0-> scheduler runs (P1=0; P2=0.5), active\_p = P2

P2->[0]

tick=1-> scheduler runs (P1=0; P2=0.5), active\_p = P2

P2->[1]

..... P2-CS2-entry ..

..... semaphore 2 locked by P2

tick=2-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[1]

..... P1-CS1-entry ..

......P1 attempted to lock semaphore 1 ..

......P1 suspended ..

......P2 inheriting PC of semaphore 2 ..

tick=3-> scheduler runs (P1=0.7; P2=1), active\_p = P2

P2->[2]

..... P2-CS1-entry ..

..... semaphore 1 locked by P2

tick=4-> scheduler runs (P1=0.7; P2=1), active\_p = P2

P2->[3]

..... P2-CS1-exit ..

......semaphore 1 unlocked by P2 ..

tick=5-> scheduler runs (P1=0.7; P2=1), active\_p = P2

P2->[4]

..... P2-CS2-exit ..

......semaphore 2 unlocked by P2 ..

......P2 priority is restored ..

......process P1 waiting on semaphore 2 is unsuspended ..

..... semaphore 1 locked by P1

tick=6-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[2]

..... P1-CS2-entry ..

..... semaphore 2 locked by P1

tick=7-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[3]

..... P1-CS2-exit ..

......semaphore 2 unlocked by P1 ..

tick=8-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[4]

..... P1-CS1-exit ..

......semaphore 1 unlocked by P1 ..

tick=9-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[5]

tick=10-> scheduler runs (P1=0.7; P2=0.5), active\_p = P1

P1->[6]

.........P1 thread ends.........

tick=11-> scheduler runs (P1=0; P2=0.5), active\_p = P2

P2->[5]

tick=12-> scheduler runs (P1=0; P2=0.5), active\_p = P2

P2->[6]

.........P2 thread ends.........

tick=13-> scheduler runs (P1=0; P2=0)

tick=14-> scheduler runs (P1=0; P2=0)

tick=15-> scheduler runs (P1=0; P2=0)

tick=16-> scheduler runs (P1=0; P2=0)

tick=17-> scheduler runs (P1=0; P2=0)

tick=18-> scheduler runs (P1=0; P2=0)

tick=19-> scheduler runs (P1=0; P2=0)

tick=20-> scheduler runs (P1=0; P2=0)

tick=21-> scheduler runs (P1=0; P2=0)

tick=22-> scheduler runs (P1=0; P2=0)

tick=23-> scheduler runs (P1=0; P2=0)

tick=24-> scheduler runs (P1=0; P2=0)

tick=25-> scheduler runs (P1=0; P2=0)

tick=26-> scheduler runs (P1=0; P2=0)

tick=27-> scheduler runs (P1=0; P2=0)

tick=28-> scheduler runs (P1=0; P2=0)

tick=29-> scheduler runs (P1=0; P2=0)