EXPERIMENT-7

NAME:SOMISH JAIN	ROLL NO:D084
SUBJECT:OPERATING SYSTEM	DATE:01/09/2025

CODE:

```
import threading
import time
import random
class PetersonLock:
    def __init__(self):
        self.flag = [False, False]
        self.turn = 0
    def acquire(self, proc_id):
        other = 1 - proc_id
        self.flag[proc_id] = True
        self.turn = other
        while self.flag[other] and self.turn == other:
            pass  # spin wait
    def release(self, proc_id):
        self.flag[proc_id] = False
def task(plock, pid):
    global shared_count, history
    for _ in range(5):
        # enter critical section
        plock.acquire(pid)
        shared_count += 3  # different increment
        history.append((pid, shared_count, time.time()))
        plock.release(pid)
        time.sleep(random.uniform(0.001, 0.003))
```

EXPERIMENT-7

```
if __name__ == "__main__":
    shared_count = 0
    history = []
    lock = PetersonLock()
    threads = [threading.Thread(target=task, args=(lock, i)) for i in (0, 1)]
    for t in threads:
        t.start()
    for t in threads:
      t.join()
    print("\n==== Run Summary ====")
    print(f"Final Shared Count = {shared_count}\n")
    for n, (proc, val, ts) in enumerate(history, 1):
        print(f"[{n:02d}] T{proc} -> {val} (at {ts:.5f})")
```

OUTPUT:

```
==== Run Summary ====
Final Shared Count = 30

[01] T0 -> 3 (at 1756745833.61489)
[02] T1 -> 6 (at 1756745833.61546)
[03] T1 -> 9 (at 1756745833.61693)
[04] T0 -> 12 (at 1756745833.61700)
[05] T1 -> 15 (at 1756745833.61872)
[06] T0 -> 18 (at 1756745833.61974)
[07] T1 -> 21 (at 1756745833.62019)
[08] T0 -> 24 (at 1756745833.62192)
[09] T1 -> 27 (at 1756745833.62270)
[10] T0 -> 30 (at 1756745833.62362)
```

EXPERIMENT-7

CODE:

```
import threading
import time
import random
class TASLock:
    def __init__(self):
        self.state = threading.Lock()
    def lock(self):
        while not self.state.acquire(blocking=False):
            continue
    def unlock(self):
        self.state.release()
def worker(lock_obj, wid):
    global total
    for _ in range(ITERATIONS):
        # acquire lock
        lock_obj.lock()
        # critical section
        total += 2   # different increment
        logs.append((wid, total))


        lock_obj.unlock()
        time.sleep(random.uniform(0.0005, 0.002))
if __name__ == "__main__":
    ITERATIONS = 6
```

EXPERIMENT-7

```
    total = 0

    logs = []

    tas_lock = TASLock()

    thread_list = [threading.Thread(target=worker, args=(tas_lock, i)) for i in
range(1, 5)]

    for th in thread_list:

        th.start()

    for th in thread_list:

        th.join()

    print("\n=== TAS Execution Trace ===")

    print(f"Final Accumulated Value: {total}\n")

    for idx, (who, val) in enumerate(logs, start=1):

        print(f"Step-{idx:02d}: W{who} updated -> {val}")
```

OUTPUT:

```
Final Accumulated Value: 48

Step-01: W1 updated -> 2
Step-02: W2 updated -> 4
Step-03: W3 updated -> 6
Step-04: W4 updated -> 8
Step-05: W4 updated -> 10
Step-06: W2 updated -> 12
Step-07: W3 updated -> 14
Step-08: W1 updated -> 16
Step-09: W2 updated -> 18
Step-10: W4 updated -> 20
Step-11: W3 updated -> 22
Step-12: W4 updated -> 24
Step-13: W2 updated -> 26
Step-14: W1 updated -> 28
Step-15: W1 updated -> 30
Step-16: W3 updated -> 32
Step-17: W2 updated -> 34
Step-18: W4 updated -> 36
Step-19: W3 updated -> 38
Step-20: W1 updated -> 40
Step-21: W2 updated -> 42
Step-22: W4 updated -> 44
Step-23: W1 updated -> 46
Step-24: W3 updated -> 48
```

EXPERIMENT-7

CODE:

```
import threading

import time

import random

# Shared state

shared_sum = 5

ROUNDS = 8   # iterations per thread

class SpinLock:

    def __init__(self):

        self._lock = threading.Lock()

    def grab(self):

        while not self._lock.acquire(blocking=False):

            time.sleep(0)  # busy wait with yield

    def drop(self):

        self._lock.release()

def execute_task(locker, worker_id):

    global shared_sum

    for _ in range(ROUNDS):

        locker.grab()

        # critical section

        shared_sum += 2   # different increment

        trace.append((worker_id, shared_sum))

    locker.drop()

        # simulate non-critical work

        time.sleep(random.uniform(0.0003, 0.0015))
```

EXPERIMENT-7

```
if __name__ == "__main__":
    trace = []
    tas_lock = SpinLock()
    workers = [threading.Thread(target=execute_task, args=(tas_lock, wid)) for wid in range(1, 5)]
    for w in workers:
        w.start()
    for w in workers:
        w.join()
    print("\n=== TAS Simulation Log ===")
    print(f"Final Result in Shared Variable: {shared_sum}\n")
    for step, (wid, val) in enumerate(trace, 1):
        print(f"#{step:02d} | Worker-{wid} set value to {val}")
```

```
Final Result in Shared Variable: 69

#01 | Worker-1 set value to 7
#02 | Worker-2 set value to 9
#03 | Worker-3 set value to 11
#04 | Worker-4 set value to 13
#05 | Worker-3 set value to 15
#06 | Worker-1 set value to 17
#07 | Worker-2 set value to 19
#08 | Worker-3 set value to 21
#09 | Worker-4 set value to 23
#10 | Worker-3 set value to 25
#11 | Worker-4 set value to 27
#12 | Worker-1 set value to 29
#13 | Worker-2 set value to 31
#14 | Worker-4 set value to 33
#15 | Worker-3 set value to 35
#16 | Worker-1 set value to 37
#17 | Worker-1 set value to 39
#18 | Worker-2 set value to 41
#19 | Worker-3 set value to 43
#20 | Worker-4 set value to 45
#21 | Worker-2 set value to 47
#22 | Worker-1 set value to 49
#23 | Worker-4 set value to 51
#24 | Worker-2 set value to 53
#25 | Worker-4 set value to 55
#26 | Worker-3 set value to 57
#27 | Worker-4 set value to 59
#28 | Worker-2 set value to 61
#29 | Worker-1 set value to 63
#30 | Worker-2 set value to 65
#31 | Worker-1 set value to 67
#32 | Worker-3 set value to 69
```

EXPERIMENT-7

CODE:

```
import threading

import time

import random


# Shared resource

accumulator = 50

CYCLES = 7   # iterations per thread

class CompareSwapLock:

    def __init__(self):

        self.flag = 0

        self.mutex = threading.Lock()


    def attempt(self, expected, newval):

        with self.mutex:

            if self.flag == expected:

                self.flag = newval

                return True

            return False


    def enter(self):

        while not self.attempt(0, 1):

            time.sleep(0)  # spin with yield


    def leave(self):
```

EXPERIMENT-7

```
        with self.mutex:
            self.flag = 0



def task(locker, wid):
    global accumulator
    for _ in range(CYCLES):
        locker.enter()

        # critical section
        accumulator += 3   # different increment
        records.append((wid, accumulator, time.time()))

        locker.leave()

        # non-critical section
        time.sleep(random.uniform(0.001, 0.003))



if __name__ == "__main__":
    records = []
    cas_lock = CompareSwapLock()

    threads = [threading.Thread(target=task, args=(cas_lock, wid)) for wid in
range(1, 4)]
```

EXPERIMENT-7

```
for t in threads:
    t.start()
for t in threads:
    t.join()


print("\n=== CAS Lock Run Report ===")
print(f"Final Accumulator Value: {accumulator}\n")


for i, (tid, val, ts) in enumerate(records, 1):
    print(f"[{i:02d}] Worker-{tid} updated total -> {val} @ {ts:.5f}")
```

OUTPUT:

```
Final Accumulator Value: 113

[01] Worker-1 updated total -> 53 @ 1756747202.53787
[02] Worker-2 updated total -> 56 @ 1756747202.53795
[03] Worker-3 updated total -> 59 @ 1756747202.53821
[04] Worker-1 updated total -> 62 @ 1756747202.53942
[05] Worker-2 updated total -> 65 @ 1756747202.54092
[06] Worker-3 updated total -> 68 @ 1756747202.54110
[07] Worker-1 updated total -> 71 @ 1756747202.54124
[08] Worker-1 updated total -> 74 @ 1756747202.54243
[09] Worker-2 updated total -> 77 @ 1756747202.54348
[10] Worker-3 updated total -> 80 @ 1756747202.54355
[11] Worker-1 updated total -> 83 @ 1756747202.54392
[12] Worker-2 updated total -> 86 @ 1756747202.54542
[13] Worker-3 updated total -> 89 @ 1756747202.54544
[14] Worker-1 updated total -> 92 @ 1756747202.54603
[15] Worker-3 updated total -> 95 @ 1756747202.54703
[16] Worker-1 updated total -> 98 @ 1756747202.54733
[17] Worker-3 updated total -> 101 @ 1756747202.54820
[18] Worker-2 updated total -> 104 @ 1756747202.54820
[19] Worker-2 updated total -> 107 @ 1756747202.55010
[20] Worker-3 updated total -> 110 @ 1756747202.55100
[21] Worker-2 updated total -> 113 @ 1756747202.55196
```