APP LAB-EX-6

Name: Rahul Goel
Reg no: RA1911030010094
Batch: CSE-O2

7)

```
import threading
import random
import time

#inheriting threading class in Thread module
class Philosopher(threading.Thread):
    running = True  #used to check if everyone is finished eating

 #Since the subclass overrides the constructor, it must make sure to invoke the base class
constructor (Thread.__init__()) before doing anything else to the thread.
    def __init__(self, index, forkOnLeft, forkOnRight):
        threading.Thread.__init__(self)
        self.index = index
        self.forkOnLeft = forkOnLeft
        self.forkOnRight = forkOnRight

    def run(self):
        while(self.running):
            # Philosopher is thinking (but really is sleeping).
            time.sleep(30)
            print ('Philosopher %s is hungry.' % self.index)
            self.dine()

    def dine(self):
        # if both the semaphores(forks) are free, then philosopher will eat
        fork1, fork2 = self.forkOnLeft, self.forkOnRight
        while self.running:
            fork1.acquire() # wait operation on left fork
            locked = fork2.acquire(False)
            if locked: break #if right fork is not available leave left fork
            fork1.release()
            print ('Philosopher %s swaps forks.' % self.index)
            fork1, fork2 = fork2, fork1
        else:
            return
        self.dining()
        #release both the fork after dining
        fork2.release()
        fork1.release()

    def dining(self):
        print ('Philosopher %s starts eating. '% self.index)
        time.sleep(30)
        print ('Philosopher %s finishes eating and leaves to think.' % self.index)

def main():
    forks = [threading.Semaphore() for n in range(5)] #initialising array of semaphore i.e forks
```

```
    #here (i+1)%5 is used to get right and left forks circularly between 1-5
    philosophers= [Philosopher(i, forks[i%5], forks[(i+1)%5])
            for i in range(5)]

    Philosopher.running = True
    for p in philosophers: p.start()
    time.sleep(100)
    Philosopher.running = False
    print ("Now we're finishing.")


if __name__ == "__main__":
    main()
```