```
from multiprocessing import Process
from multiprocessing import Condition, Lock
from multiprocessing import Array, Manager
import time
import random

from monitor import Table

NPHIL = 5
K = 100


def delay(n):
    time.sleep(random.random()/n)

def philosopher_task(num:int, table: Table):
    table.set_current_phil(num)
    while True:
        print (f"Philosofer {num} thinking")
        print (f"Philosofer {num} wants to eat")
        table.wants_eat(num)
        print (f"Philosofer {num} eating")
        table.wants_think(num)
        print (f"Philosofer {num} stops eating")

def main():
    manager = Manager()
    table = Table(NPHIL, manager)
    philosofers = [Process(target=philosopher_task, args=(i,table)) \
                   for i in range(NPHIL)]
    for i in range(NPHIL):
        philosofers[i].start()
    for i in range(NPHIL):
        philosofers[i].join()

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

```
from multiprocessing import Process
from multiprocessing import Condition, Semaphore, Lock
from multiprocessing import Array, Manager, Value
import time
import random
from monitor import Table, CheatMonitor

NPHIL = 5
K = 100




def delay(n):
    time.sleep(random.random()/n)

def philosopher_task(num:int, table: Table, cheat: CheatMonitor):
    table.set_current_phil(num)
    while True:
        print (f"Philosofer {num} thinking")
        print (f"Philosofer {num} wants to eat")
        table.wants_eat(num)
        if num == 0 or num == 2:
            cheat.is_eating(num)
        print(f"Philosofer {num} eating")
        if num == 0 or num == 2:
            cheat.wants_think(num)
        table.wants_think(num)
        print (f"Philosofer {num} stops eating")

def main():
    manager = Manager()
    table = Table(NPHIL, manager)
    cheat = CheatMonitor()
    philosofers = [Process(target=philosopher_task,
                           args=(i, table,
                                 cheat)) \
                   for i in range(NPHIL)]
    for i in range(NPHIL):
        philosofers[i].start()
    for i in range(NPHIL):
        philosofers[i].join()

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

```
from monitor import AnticheatTable as Table, CheatMonitor
from multiprocessing import Process, Manager
import time
import random

NPHIL = 5

def delay(n):
    time.sleep(random.random()/n)

def philosopher_task(num:int, table: Table, cheat: CheatMonitor):
    table.set_current_phil(num)
    cont = 0
    while cont<=100:
        print (f"Philosofer {num} thinking {cont}")
        print (f"Philosofer {num} wants to eat {cont}")
        table.wants_eat(num)
        if num == 0 or num == 2:
            cheat.is_eating(num)
        print(f"Philosofer {num} eating {cont}")
        if num == 0 or num == 2:
            cheat.wants_think(num)
        table.wants_think(num)
        print (f"Philosofer {num} stops eating {cont}")
        cont += 1

def main():
    manager = Manager()
    table = Table(NPHIL, manager)
    cheat = CheatMonitor()
    philosofers = [Process(target=philosopher_task, args=(i,table, cheat)) \
                   for i in range(NPHIL)]
    for i in range(NPHIL):
        philosofers[i].start()
    for i in range(NPHIL):
        philosofers[i].join()

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