# Assignment 2: Dining Philosophers Question A2.1

In [2]:
```
import threading
import time
from pynq.overlays.base import BaseOverlay
from pynq.lib import RGBLED

base = BaseOverlay("base.bit")
```

In [3]:
```
led4 = RGBLED(4)

def blink(t, d, n):
    for _ in range(t):
        if stop_event.is_set():
            break
        if n < 4:
            base.leds[n].toggle()
        else:
            led4.write(0x2)
        time.sleep(d)
        if n < 4:
            base.leds[n].toggle()
        else:
            led4.write(0x0)
        time.sleep(d)
    turn_off(n)

def turn_off(n):
    if n < 4:
        base.leds[n].off()
    else:
        led4.write(0x0)

stop_event = threading.Event()

def philosopher(_l_fork_left, _l_fork_right, num):
    '''
    _l_fork_left: threading lock for the left fork
    _l_fork_right: threading lock for the right fork
    num: philosopher number (also used for LED index)
    '''
    while not stop_event.is_set():
        turn_off(num)
        print(f"Philosopher {num} is starving.")

        fork_left_acquired = _l_fork_left.acquire(False)
        fork_right_acquired = _l_fork_right.acquire(False)

        if fork_left_acquired and fork_right_acquired:
            print(f"Philosopher {num} is eating.")
            blink(5, 0.1, num)
            _l_fork_left.release()
            _l_fork_right.release()
            print(f"Philosopher {num} is napping.")
            blink(3, 1.0, num)
        else:
            if fork_left_acquired:
                _l_fork_left.release()
            if fork_right_acquired:
                _l_fork_right.release()
            time.sleep(1.0)

        # prevent starvation
        time.sleep(0.1)
```

In [4]:
```
def philosopher(_l_fork_left, _l_fork_right, num):
    '''
    _l_fork_left: threading lock for the left fork
    _l_fork_right: threading lock for the right fork
    num: philosopher number (also used for LED index)
    '''
    while not stop_event.is_set():

        turn_off(num)
        print(f"Philosopher {num} is starving.")

        fork_left_acquired = _l_fork_left.acquire(False)
        fork_right_acquired = _l_fork_right.acquire(False)

        if fork_left_acquired and fork_right_acquired:

            print(f"Philosopher {num} is eating.")
            blink(5, 0.1, num)

            _l_fork_left.release()
            _l_fork_right.release()


            print(f"Philosopher {num} is napping.")
            blink(3, 1.0, num)
        else:
            if fork_left_acquired:
                _l_fork_left.release()
            if fork_right_acquired:
                _l_fork_right.release()
            time.sleep(1.0)

        time.sleep(0.1)
```

In [5]:
```
def monitor_buttons():
    while not stop_event.is_set():
        if base.buttons.read() != 0:
            print("Button pressed. Shutting down.")
            stop_event.set()
            # Turn off all LEDs
            for i in range(4):
                base.leds[i].off()
            led4.write(0x0)
            break
        time.sleep(0.1)

def main():
    forks = [threading.Lock() for _ in range(5)]

    philosophers = []
    for i in range(5):
        left_fork = forks[i]
        right_fork = forks[(i + 1) % 5]
        t = threading.Thread(target=philosopher, args=(left_fork, right_fork, i))
        philosophers.append(t)
        t.start()

    button_thread = threading.Thread(target=monitor_buttons)
    button_thread.start()

    for t in philosophers:
        t.join()
    button_thread.join()
    print("Program terminated.")

main()
```

```
Philosopher 0 is starving.
Philosopher 0 is eating.
Philosopher 1 is starving.
Philosopher 2 is starving.
Philosopher 2 is eating.
Philosopher 3 is starving.
Philosopher 4 is starving.
Philosopher 0 is napping.
Philosopher 2 is napping.
Philosopher 1 is starving.
Philosopher 1 is eating.
Philosopher 3 is starving.
Philosopher 3 is eating.
Philosopher 4 is starving.
Philosopher 1 is napping.
Philosopher 3 is napping.
Philosopher 4 is starving.
Philosopher 4 is eating.
Philosopher 4 is napping.
Philosopher 0 is starving.
Philosopher 0 is eating.
Philosopher 2 is starving.
Philosopher 2 is eating.
Philosopher 0 is napping.
Philosopher 2 is napping.
Philosopher 1 is starving.
Philosopher 1 is eating.
Philosopher 3 is starving.
Philosopher 3 is eating.
Philosopher 1 is napping.
Philosopher 3 is napping.
Philosopher 4 is starving.
Philosopher 4 is eating.
Philosopher 4 is napping.
Philosopher 0 is starving.
Philosopher 0 is eating.
Philosopher 2 is starving.
Philosopher 2 is eating.
Button pressed. Shutting down.
Philosopher 0 is napping.
Philosopher 2 is napping.
Program terminated.
```

```
In [ ]:
```