In [ ]:

```python
#OS Lab Assignment on Semaphores
#Submitted by: Sanskar Sharma
#PRN: 0120180381
#Roll number: 090
#Date: 15/10/2020


#OS Lab Assignment:
#Part1: Process Synronization Using Semaphores

# Python program to illustrate
# the concept of locks
# in multiprocessing
import multiprocessing

# function to withdraw from account
def withdraw(balance, lock):
    for _ in range(10000):
        lock.acquire()
        balance.value = balance.value - 1
        lock.release()

# function to deposit to account
def deposit(balance, lock):
    for _ in range(10000):
        lock.acquire()
        balance.value = balance.value + 1
        lock.release()

def perform_transactions():

    # initial balance (in shared memory)
    balance = multiprocessing.Value('i', 100)

    # creating a lock object
    lock = multiprocessing.Lock()

    # creating new processes
    p1 = multiprocessing.Process(target=withdraw, args=(balance,lock))
    p2 = multiprocessing.Process(target=deposit, args=(balance,lock))

    # starting processes
    p1.start()
    p2.start()

    # wait until processes are finished
    p1.join()
    p2.join()

    # print final balance
    print("Final balance = {}".format(balance.value))


# function to withdraw from account
def withdraw_withoutS(balance):
    for _ in range(10000):
        balance.value = balance.value - 1

# function to deposit to account
def deposit_withoutS(balance):
    for _ in range(10000):
        balance.value = balance.value + 1

def perform_transactions_withoutS():

    # initial balance (in shared memory)
    balance = multiprocessing.Value('i', 100)

    # creating new processes
    p1 = multiprocessing.Process(target=withdraw_withoutS, args=(balance,))
    p2 = multiprocessing.Process(target=deposit_withoutS, args=(balance,))

    # starting processes
    p1.start()
    p2.start()

    # wait until processes are finished
    p1.join()
    p2.join()

    # print final balance
    print("Final balance = {}".format(balance.value))

if __name__ == "__main__":
    print("\t\t\t\tOS Lab Assignment by Sanskar Sharma")
    print("\t\t\t  Part 1: Process Synchronization using Semaphore")
    while(1):
        print("Enter\n1. Process Synchronization without using Semaphore.\n2. Process Synchr
onization using Semaphores")
        print("3. Exit")
        val=input("Enter your choice: ")
        if (int(val)==1):
            for _ in range(10):
                # perform same transaction process 10 times without semaophore
                #Race condition occurs
                perform_transactions_withoutS()
        elif (int(val)==2):
            for _ in range(10):
                # perform same transaction process 10 times using semaphore
                #handles race conditions
                perform_transactions()
        else:
            break

"""
Output:

runfile('C:/Users/hp/Desktop/ProcessSynchronisation.py', wdir='C:/Users/hp/Desktop')
                                OS Lab Assignment by Sanskar Sharma
                          Part 1: Process Synchronization using Semaphore
Enter
1. Process Synchronization without using Semaphore.
2. Process Synchronization using Semaphores
3. Exit

Enter your choice: 1
Final balance = 5600
Final balance = 1660
Final balance = -6786
Final balance = 6056
Final balance = 1802
Final balance = -182
Final balance = 189
Final balance = -1042
Final balance = 724
Final balance = 6338
Enter
1. Process Synchronization without using Semaphore.
2. Process Synchronization using Semaphores
3. Exit

Enter your choice: 2
Final balance = 100
Final balance = 100
Final balance = 100
Final balance = 100
Final balance = 100
Final balance = 100
Final balance = 100
Final balance = 100
Final balance = 100
Final balance = 100
Enter
1. Process Synchronization without using Semaphore.
2. Process Synchronization using Semaphores
3. Exit

Enter your choice: 3

"""

#OS Lab Assignment:
#Part 2: Readers and Writers Problem using Semaphore

import threading
import time

class ReaderWriter():
    def __init__(self):
        self.rd = threading.Semaphore()  #initializing semaphores using Semaphore class in
                                         # threading module for reading and wrting
        self.wrt = threading.Semaphore()

        self.readCount = 0   #initializing number of reader present

    def reader(self):
        while True:
            self.rd.acquire()      #wait on read semaphore

            self.readCount+=1       #increase count for reader by 1

            if self.readCount == 1: #since reader is present, prevent writing on data
                self.wrt.acquire() #wait on write semaphore

            self.rd.release()     #sinal on read semaphore

            print(f"Reader {self.readCount} is reading")

            self.rd.acquire()   #wait on read semaphore

            self.readCount-=1   #reading performed by reader hence decrementing readercount

            if self.readCount == 0: #if no reader is present allow writer to write the data
                self.wrt.release()  # signal on write semphore, now writer can write

            self.rd.release()      #sinal on read semaphore

            time.sleep(3)

    def writer(self):
        while True:
            self.wrt.acquire()     #wait on write semaphore

            print("Wrting data.....")  # write the data
            print("-"*20)

            self.wrt.release()      #sinal on write semaphore

            time.sleep(3)

    def main(self):
        # calling mutliple readers and writers
        t1 = threading.Thread(target = self.reader)
        t1.start()
        t2 = threading.Thread(target = self.writer)
        t2.start()
        t3 = threading.Thread(target = self.reader)
        t3.start()
        t4 = threading.Thread(target = self.reader)
        t4.start()
        t6 = threading.Thread(target = self.writer)
        t6.start()
        t5 = threading.Thread(target = self.reader)
        t5.start()


if __name__=="__main__":
    print("\t\t\t\tOS Lab Assignment by Sanskar Sharma")
    print("\t\t\t  Part 2: Readers and Writers Problem using Semaphore")
    c = ReaderWriter()
    c.main()

"""
Output: Sequence of Writing and Reader's reading!
Multiple reader's can also read at the same time but only one writes!

                                OS Lab Assignment by Sanskar Sharma
                          Part 2: Readers and Writers Problem using Semaphore

Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is reading
Wrting data.....
--------------------
Reader 1 is readingReader 2 is reading

Reader 1 is readingReader 2 is reading

Wrting data.....
--------------------
Wrting data.....
--------------------
Reader 1 is readingReader 2 is reading

Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Wrting data.....
--------------------
Reader 1 is readingReader 2 is reading

Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Wrting data.....
--------------------
Reader 1 is readingReader 2 is reading

Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Wrting data.....
--------------------
Reader 1 is readingReader 2 is reading

Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Wrting data.....
--------------------
Reader 1 is readingReader 2 is reading

Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------
Wrting data.....
--------------------
Reader 1 is readingReader 2 is reading

Reader 1 is reading
Reader 1 is reading
Wrting data.....
--------------------

(used keyboard interrupt to exit!!)
"""
```