/// spos practical 4  
  
  
import threading

import time

import random

# Producer-Consumer Problem

def producer\_consumer():

    buffer = []

    buffer\_size = int(input("Enter buffer size for Producer-Consumer: "))

    max\_items = int(input("Enter the number of items to produce and consume: "))

    buffer\_lock = threading.Lock()

    empty = threading.Semaphore(buffer\_size)

    full = threading.Semaphore(0)

    def producer():

        for \_ in range(max\_items):

            item = random.randint(1, 100)

            empty.acquire()

            buffer\_lock.acquire()

            buffer.append(item)

            print(f"Producer produced: {item}")

            buffer\_lock.release()

            full.release()

            time.sleep(random.random())

    def consumer():

        for \_ in range(max\_items):

            full.acquire()

            buffer\_lock.acquire()

            if buffer:

                item = buffer.pop(0)

                print(f"Consumer consumed: {item}")

            buffer\_lock.release()

            empty.release()

            time.sleep(random.random())

    producer\_thread = threading.Thread(target=producer)

    consumer\_thread = threading.Thread(target=consumer)

    producer\_thread.start()

    consumer\_thread.start()

    producer\_thread.join()

    consumer\_thread.join()

# Dining Philosophers Problem

def dining\_philosophers():

    class Philosopher(threading.Thread):

        def \_\_init\_\_(self, index, left\_fork, right\_fork, iterations):

            threading.Thread.\_\_init\_\_(self)

            self.index = index

            self.left\_fork = left\_fork

            self.right\_fork = right\_fork

            self.iterations = iterations

        def run(self):

            for \_ in range(self.iterations):

                print(f"Philosopher {self.index} is thinking.")

                time.sleep(random.random())

                print(f"Philosopher {self.index} is hungry.")

                self.eat()

        def eat(self):

            with self.left\_fork:

                with self.right\_fork:

                    print(f"Philosopher {self.index} is eating.")

                    time.sleep(random.random())

                    print(f"Philosopher {self.index} finished eating.")

    forks = [threading.Lock() for \_ in range(5)]

    iterations = int(input("Enter number of iterations for each philosopher: "))

    philosophers = [Philosopher(i, forks[i % 5], forks[(i + 1) % 5], iterations) for i in range(5)]

    for p in philosophers:

        p.start()

    for p in philosophers:

        p.join()

# Readers-Writers Problem

def readers\_writers():

    mutex = threading.Lock()

    write\_block = threading.Lock()

    readers\_count = 0

    read\_operations = int(input("Enter number of read operations: "))

    write\_operations = int(input("Enter number of write operations: "))

    def reader():

        nonlocal readers\_count

        for \_ in range(read\_operations):

            mutex.acquire()

            readers\_count += 1

            if readers\_count == 1:

                write\_block.acquire()

            mutex.release()

            print(f"Reader {threading.current\_thread().name} is reading.")

            time.sleep(1)

            print(f"Reader {threading.current\_thread().name} finished reading.")

            mutex.acquire()

            readers\_count -= 1

            if readers\_count == 0:

                write\_block.release()

            mutex.release()

            time.sleep(1)

    def writer():

        for \_ in range(write\_operations):

            print(f"Writer {threading.current\_thread().name} is waiting to write.")

            write\_block.acquire()

            print(f"Writer {threading.current\_thread().name} is writing.")

            time.sleep(2)

            print(f"Writer {threading.current\_thread().name} finished writing.")

            write\_block.release()

            time.sleep(1)

    for i in range(3):

        threading.Thread(target=reader, name=f'Reader-{i+1}').start()

    for i in range(2):

        threading.Thread(target=writer, name=f'Writer-{i+1}').start()

# Main Function with Menu (Switch Case)

def main():

    while True:

        print("\n--- Synchronization Problems Menu ---")

        print("1. Producer-Consumer Problem")

        print("2. Dining Philosophers Problem")

        print("3. Readers-Writers Problem")

        print("4. Exit")

        choice = int(input("Choose a problem to run (1-4): "))

        if choice == 1:

            print("\nRunning Producer-Consumer Problem...")

            producer\_consumer()

        elif choice == 2:

            print("\nRunning Dining Philosophers Problem...")

            dining\_philosophers()

        elif choice == 3:

            print("\nRunning Readers-Writers Problem...")

            readers\_writers()

        elif choice == 4:

            print("\nExiting...")

            break

        else:

            print("Invalid choice. Please select a valid option.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()  
  
  
//output  
--- Synchronization Problems Menu ---

1. Producer-Consumer Problem

2. Dining Philosophers Problem

3. Readers-Writers Problem

4. Exit

Choose a problem to run (1-4): 1

Running Producer-Consumer Problem...

Enter buffer size for Producer-Consumer: 5

Enter the number of items to produce and consume: 10

Producer produced: 61

Consumer consumed: 61

Producer produced: 96

Consumer consumed: 96

Producer produced: 32

Consumer consumed: 32

Producer produced: 34

Consumer consumed: 34

Producer produced: 1

Consumer consumed: 1

Producer produced: 6

Consumer consumed: 6

Producer produced: 85

Consumer consumed: 85

Producer produced: 63

Producer produced: 21

Producer produced: 27

Consumer consumed: 63

Consumer consumed: 21

Consumer consumed: 27

--- Synchronization Problems Menu ---

1. Producer-Consumer Problem

2. Dining Philosophers Problem

3. Readers-Writers Problem

4. Exit

Choose a problem to run (1-4): 2

Running Dining Philosophers Problem...

Enter number of iterations for each philosopher: 5

Philosopher 0 is thinking.

Philosopher 1 is thinking.

Philosopher 2 is thinking.

Philosopher 3 is thinking.

Philosopher 4 is thinking.

Philosopher 4 is hungry.

Philosopher 4 is eating.

Philosopher 4 finished eating.

Philosopher 4 is thinking.

Philosopher 0 is hungry.

Philosopher 0 is eating.

Philosopher 4 is hungry.

Philosopher 3 is hungry.

Philosopher 2 is hungry.

Philosopher 0 finished eating.

Philosopher 0 is thinking.

Philosopher 4 is eating.

Philosopher 0 is hungry.

Philosopher 1 is hungry.

Philosopher 4 finished eating.

Philosopher 4 is thinking.

Philosopher 3 is eating.

Philosopher 3 finished eating.

Philosopher 3 is thinking.

Philosopher 2 is eating.

Philosopher 3 is hungry.

Philosopher 4 is hungry.

Philosopher 2 finished eating.

Philosopher 2 is thinking.

Philosopher 1 is eating.

Philosopher 1 finished eating.

Philosopher 1 is thinking.

Philosopher 0 is eating.

Philosopher 2 is hungry.

Philosopher 0 finished eating.

Philosopher 0 is thinking.

Philosopher 4 is eating.

Philosopher 1 is hungry.

Philosopher 4 finished eating.

Philosopher 4 is thinking.

Philosopher 3 is eating.

Philosopher 0 is hungry.

Philosopher 4 is hungry.

Philosopher 3 finished eating.

Philosopher 3 is thinking.

Philosopher 2 is eating.

Philosopher 3 is hungry.

Philosopher 2 finished eating.

Philosopher 2 is thinking.

Philosopher 1 is eating.

Philosopher 1 finished eating.

Philosopher 1 is thinking.

Philosopher 0 is eating.

Philosopher 2 is hungry.

Philosopher 1 is hungry.

Philosopher 0 finished eating.

Philosopher 0 is thinking.

Philosopher 4 is eating.

Philosopher 4 finished eating.

Philosopher 4 is thinking.

Philosopher 3 is eating.

Philosopher 0 is hungry.

Philosopher 4 is hungry.

Philosopher 3 finished eating.

Philosopher 3 is thinking.

Philosopher 2 is eating.

Philosopher 3 is hungry.

Philosopher 2 finished eating.

Philosopher 2 is thinking.

Philosopher 1 is eating.

Philosopher 1 finished eating.

Philosopher 1 is thinking.

Philosopher 0 is eating.

Philosopher 1 is hungry.

Philosopher 2 is hungry.

Philosopher 0 finished eating.

Philosopher 0 is thinking.

Philosopher 4 is eating.

Philosopher 0 is hungry.

Philosopher 4 finished eating.

Philosopher 3 is eating.

Philosopher 3 finished eating.

Philosopher 3 is thinking.

Philosopher 2 is eating.

Philosopher 2 finished eating.

Philosopher 2 is thinking.

Philosopher 1 is eating.

Philosopher 2 is hungry.

Philosopher 3 is hungry.

Philosopher 3 is eating.

Philosopher 3 finished eating.

Philosopher 1 finished eating.

Philosopher 1 is thinking.

Philosopher 2 is eating.

Philosopher 0 is eating.

Philosopher 0 finished eating.

Philosopher 2 finished eating.

Philosopher 1 is hungry.

Philosopher 1 is eating.

Philosopher 1 finished eating.

--- Synchronization Problems Menu ---

1. Producer-Consumer Problem

2. Dining Philosophers Problem

3. Readers-Writers Problem

4. Exit

Choose a problem to run (1-4): 3

Running Readers-Writers Problem...

Enter number of read operations: 5

Enter number of write operations: 3

Reader Reader-1 is reading.

Reader Reader-2 is reading.

Reader Reader-3 is reading.

Writer Writer-1 is waiting to write.

Writer Writer-2 is waiting to write.

--- Synchronization Problems Menu ---

1. Producer-Consumer Problem

2. Dining Philosophers Problem

3. Readers-Writers Problem

4. Exit

Choose a problem to run (1-4): Reader Reader-2 finished reading.

Reader Reader-1 finished reading.

Reader Reader-3 finished reading.

Writer Writer-1 is writing.

Writer Writer-1 finished writing.

Writer Writer-2 is writing.

Writer Writer-1 is waiting to write.

Writer Writer-2 finished writing.

Reader Reader-1 is reading.

Reader Reader-3 is reading.

Reader Reader-2 is reading.

Writer Writer-2 is waiting to write.

Reader Reader-1 finished reading.

Reader Reader-3 finished reading.

Reader Reader-2 finished reading.

Writer Writer-1 is writing.

Writer Writer-1 finished writing.

Writer Writer-2 is writing.

Writer Writer-1 is waiting to write.

Writer Writer-2 finished writing.

Reader Reader-1 is reading.

Reader Reader-3 is reading.

Reader Reader-2 is reading.

Writer Writer-2 is waiting to write.

Reader Reader-1 finished reading.

Reader Reader-3 finished reading.

Reader Reader-2 finished reading.

Writer Writer-1 is writing.

Writer Writer-1 finished writing.

Writer Writer-2 is writing.

Writer Writer-2 finished writing.

Reader Reader-2 is reading.

Reader Reader-1 is reading.

Reader Reader-3 is reading.

Reader Reader-2 finished reading.

Reader Reader-1 finished reading.

Reader Reader-3 finished reading.

Reader Reader-2 is reading.

Reader Reader-1 is reading.

Reader Reader-3 is reading.

Reader Reader-2 finished reading.

Reader Reader-1 finished reading.

Reader Reader-3 finished reading.