Exercises

1Write a python program that demonstrates the synchronization of Readers and Writer Problem using Semaphore.

Ans:

semaphore.py:

|  |
| --- |
| from threading import Semaphore, Thread  import time  import random  # Shared resource  shared\_resource = 0  # Synchronization primitives  resource\_semaphore = Semaphore(1) # Controls access to shared resource  reader\_count\_lock = Semaphore(1) # Protects reader\_count  reader\_count = 0 # Tracks active readers  # Reader function  def reader(reader\_id):  global reader\_count  print(f"Reader {reader\_id} wants to read")  with reader\_count\_lock:  reader\_count += 1  if reader\_count == 1:  resource\_semaphore.acquire() # First reader locks resource  print(f"Reader {reader\_id} reading: {shared\_resource}")  time.sleep(random.uniform(0.1, 0.3)) # Simulate reading  with reader\_count\_lock:  reader\_count -= 1  if reader\_count == 0:  resource\_semaphore.release() # Last reader releases resource  print(f"Reader {reader\_id} done reading")  # Writer function  def writer(writer\_id):  global shared\_resource  print(f"Writer {writer\_id} wants to write")  with resource\_semaphore:  print(f"Writer {writer\_id} writing")  shared\_resource += 1  time.sleep(random.uniform(0.2, 0.5)) # Simulate writing  print(f"Writer {writer\_id} wrote: {shared\_resource}")  # Main function  def main():  threads = []  for i in range(4): # 4 readers  threads.append(Thread(target=reader, args=(i,)))  for i in range(2): # 2 writers  threads.append(Thread(target=writer, args=(i,)))    random.shuffle(threads) # Randomize execution order  for thread in threads:  thread.start()  for thread in threads:  thread.join()  if \_\_name\_\_ == "\_\_main\_\_":  main() |

OUTPUT:

