NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES
FAST - PESHAWAR CAMPUS

Subject: Software Construction and Development Lab (CL-3001)

Instructor: M. Saood Sarwar

Quiz : 3 **Section: B**

Name: _____ **Roll no. _____**

Question 1: The code below demonstrates concurrency combined with exception handling. Analyze it carefully and predict the output. If there will be an error in the code write **ERROR** in output section**.**

Code	Output

import asyncio
import threading
shared_list = [10, 20, 30, 40]
def blocking_sum():
  global shared_list
  total = sum(shared_list)
  print(f"Blocking Sum: {shared_list} is {total}")
  shared_list.append(total)
 	
async def async_average():
  global shared_list
  await asyncio.sleep(3)
  if not shared_list:
  print("Async Error: No numbers provided.")
  else:
  avg = sum(shared_list) / len(shared_list)
  print(f"Async Average: {shared_list} is {avg}")
 	
async def main():
  global shared_list
  thread = threading.Thread(target=blocking_sum)
  thread.start()
  shared_list = [50, 60, 70]
  await async_average()
  thread.join()
await main()	

Code	Output

import threading
total_conversions = 0
def currency_conversion_tracker(amount, rate):
  global total_conversions
  try:
  if rate <= 0:
  raise ValueError("Exchange rate must be positive.")
  converted = amount * rate
  print(f"Converted {amount} at rate {rate} is {converted}")
 	
  total_conversions += converted
  except BaseException as e:
  print(f"Error: {e}")
  except Exception as e:
  print(f"Error: {e}")
 	
amounts = [60, 120, 180, 240]
rates = [1, -0.5, 0, 1.5]
 	
threads = [threading.Thread(target=currency_conversion_tracker, args=(amt, rate)) for amt, rate in zip(amounts, rates)]
 	
for thread in threads:
  thread.start()
 	
for thread in threads:
  thread.join()
print("Final total conversions:", total_conversions)	