PP - FINAL MCQ QUESTION BANK

Helping Others Have Special taste

Questions

1. What is the key drawback of overusing locks in parallel programs?
A) Increased memory usage
B) High CPU load
C) Reduced concurrency
D) More IO waiting

2. What distinguishes parallel from serial operations at the most fundamental level?
A) The use of different programming languages
B) The way registers are utilized
C) The type of data being processed
D) The speed of individual CPUs

3. What are multi-core processors?
A) Chips with only one CPU
B) Integrated circuits with two or more CPUs
C) Processors designed only for graphical computations
D) Processors with sequential processing capabilities

4. How does parallel processing enhance real-time data handling?
A) By using fine-grained parallelism for frequent communication among tasks
B) By slowing down task communication for accuracy
C) By relying entirely on a single CPU
D) By skipping intermediate computation steps

5. What makes clusters unique in parallel computing?
A) They are single powerful CPUs used for computations
B) They are groups of commercial computers linked by a network
C) They are processors designed solely for graphics
D) They use massively parallel processors for computations

6. What does the control unit in a SIMD system do?
A) It assigns individual tasks to processors
B) It processes all data sequentially
C) It provides identical instructions to all processors
D) It performs all computations independently

7. Execution of several activities at the same time.
a) processing
b) parallel processing
c) serial processing
d) multitasking

8. Parallel processing has single execution flow.
a) True
b) False

9. A term for simultaneous access to a resource, physical or logical.
a) Multiprogramming
b) Multitasking
c) Threads
d) Concurrency

10. ______ leads to concurrency.
a) Serialization
b) Parallelism
c) Serial processing
d) Distribution

11. Several instructions execution simultaneously in _________
a) processing mm
b) parallel processing
c) serial processing
d) multitasking

12. Multiprocessing allows single processor to run several concurrent threads.
a) True
b) False

13. A parallel computing system consists of multiple processor that communicate with each other using a ___.
A) Allocated memory
B) Shared memory
C) Network
D) None of the above

14. In parallel computing systems, as the number of processors increases, with enough parallelism available in applications.
A) True
B) False

15. Parallel computing can be used in ___
A) Science and engineering
B) Database and data mining
C) Real time simulation of systems
D) All of the above

16. What is the primary goal of parallel programming?
A) To reduce code complexity
B) To enhance performance and efficiency
C) To simplify debugging
D) To increase memory usage

17. What type of parallelism involves executing different tasks concurrently?
A) Data parallelism
B) Task parallelism
C) Sequential programming
D) Synchronous programming

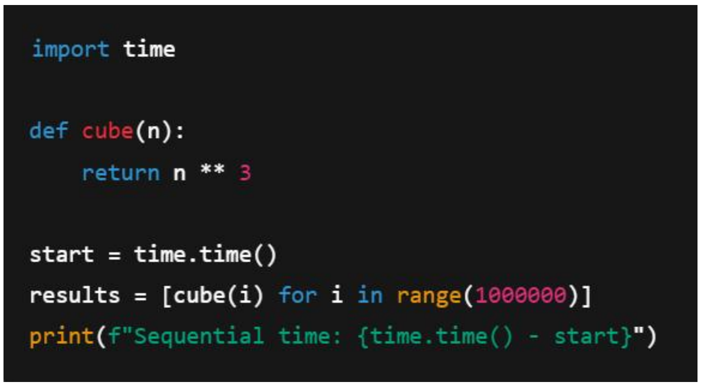
18. Which type of parallelism divides data into smaller chunks?
A) Task parallelism
B) Data parallelism
C) Process parallelism
D) Thread parallelism

19. What is the Global Interpreter Lock (GIL) primarily a limitation for?
A) I/O-bound tasks
B) CPU-bound tasks
C) Networking tasks
D) Memory-bound tasks

20. Which Python library is suitable for I/O-bound tasks?
A) multiprocessing
B) concurrent.futures
C) threading
D) numpy

21. What is the main advantage of multiprocessing over threading in Python?
A) Simpler code
B) Bypasses the GIL
C) Easier debugging
D) More readable syntax

22. Answer the following questions (i–iv) based on following code:



```python
import time

def cube(n):
    return n ** 3

start = time.time()
results = [cube(i) for i in range(1000000)]
print(f"Sequential time: {time.time() - start}")
```

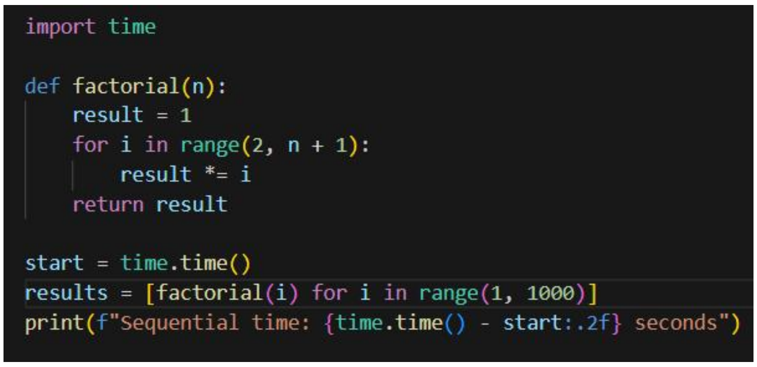
i. What does the function cube(n) do?
a) Calculates the square root of n
b) Calculates the cube of n
c) Calculates the factorial of n
d) Divides n by 3

ii. What is the nature of execution in the provided code?
a) Parallel
b) Multithreaded
c) Sequential
d) Distributed

iii. Why might this sequential execution be suboptimal on a modern multi-core CPU?
a) It uses too many threads
b) It uses multiprocessing instead of multithreading
c) It only uses a single core, leaving others idle
d) It creates too many processes

iv. What would be a benefit of parallelizing this code using multiprocessing?
a) It would reduce CPU usage
b) It would run slower but use less memory
c) It could run faster by utilizing multiple CPU cores
d) It would make the result more accurate

23. What kind of task is this and how could it be optimized?
a) I/O-bound, using less memory
b) CPU-bound, using multiprocessing to split the range
c) Memory-bound, using generators
d) GPU-bound, moving it to CUDA



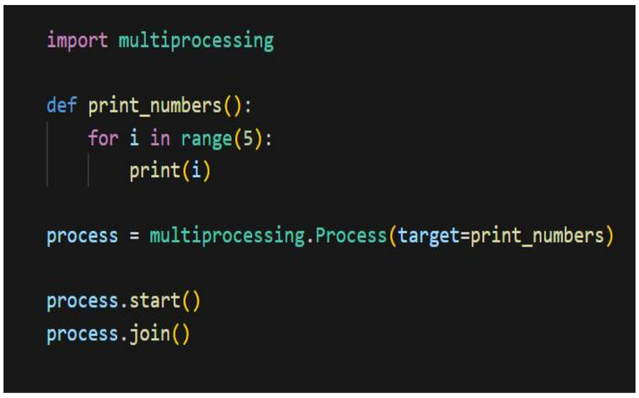
```python
import time

def factorial(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

start = time.time()
results = [factorial(i) for i in range(1, 1000)]
print(f"Sequential time: {time.time() - start:.2f} seconds")
```

24. What is the drawback of sequential programming?
A) Inefficient for large tasks
B) Hard to implement
C) Complex syntax
D) Limited readability

25. In the example, what does the multiprocessing module utilize?
A) Multiple threads
B) Multiple cores
C) Single-core execution
D) Networked processes



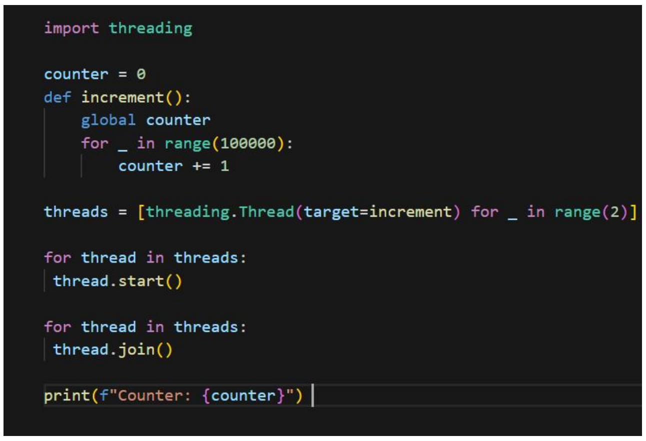
```python
import multiprocessing

def print_numbers():
    for i in range(5):
        print(i)

process = multiprocessing.Process(target=print_numbers)

process.start()
process.join()
```

26. What is the result of executing the threading example provided below?
A) Improved CPU-bound performance
B) May not achieve expected counter value
C) Counter always reaches 200,000
D) Faster execution due to GIL



```python
import threading

counter = 0
def increment():
    global counter
    for _ in range(100000):
        counter += 1

threads = [threading.Thread(target=increment) for _ in range(2)]

for thread in threads:
 thread.start()

for thread in threads:
 thread.join()

print(f"Counter: {counter}")
```

27. What happens when CPU-bound tasks are executed with threading in Python?
A) True parallelism is achieved
B) Limited performance due to GIL
C) Faster execution than multiprocessing
D) Multiple threads run at a time

28. Why is multiprocessing more effective for CPU-bound tasks?
A) Simpler than threading
B) Utilizes multiple CPU cores
C) Automatically handles I/O-bound tasks
D) Prevents data chunking

29. What is the primary limitation of sequential programming?
A) Requires advanced hardware
B) Cannot handle large datasets
C) Cannot utilize modern multi-core processors effectively
D) Poor readability

30. How can the GIL affect multi-threaded CPU-bound tasks?
A) Allows true parallelism
B) Improves performance
C) Limits performance
D) Automatically manages resources

31. Which statement best describes task parallelism?
A) Performing the same operation on all data chunks concurrently
B) Dividing tasks into smaller units for simultaneous execution
C) Using a single thread for all tasks
D) Processing tasks sequentially

32. What is the purpose of the join() method in threading?
A) To start a thread
B) To terminate a thread
C) To wait for a thread to finish execution
D) To create a new thread

33. In Python, what is the main advantage of using multiprocessing over threading?
A) Works seamlessly with the GIL
B) Does not require chunking of data
C) Supports true parallelism
D) Best suited for I/O-bound tasks

34. Why do threads share the same memory space?

A) to avoid memory leaks

B) to make communication between threads more efficient

C) to make them faster than processes

D) to increase CPU load

35. What does asynchronous programming primarily allow?

A) Multi-threading

B) Non-blocking operations

C) Real-time execution

D) Sequential processing

36. What makes debugging parallel programs more difficult than sequential programs?

A) More code

B) Network issues

C) Race conditions and synchronization issues

D) Compiler errors

37. Which tool helps identify bottlenecks in parallel Python code?

A) Task manager

B) PyLint

C) cProfile

D) DebugPy

38. Which threading issue can occur even if there is no deadlock?

A) Starvation

B) Syntax error

C) Infinite loop

D) Memory leak

39. What is the function of massively parallel processing (MPP)?

A) Simplify small-scale computations

B) Process immense datasets rapidly

C) Restrict communication between processors

D) Operate without messaging interface

40. Which is better for CPU-bound tasks, threading or multiprocessing?

A) threading

B) multiprocessing

C) both are the same

D) neither is useful

41. What is the main use case for multiprocessing?
A) running CPU-intensive tasks in parallel
B) running multiple threads in parallel
C) running simple print statements
D) running a single-threaded server

42. Why use queues in multiprocessing?
A) to avoid shared memory conflicts
B) to store large amounts of data
C) to increase CPU speed
D) to replace threading

43. Which keyword is used to define a coroutine?
A) coroutine
B) async def
C) def async
D) await def

44. What does asyncio.create_task() do?
A) Runs a coroutine immediately
B) Blocks all tasks
C) Creates a new thread
D) Schedules a coroutine to run

45. What is a multiprocessing.Value used for?
A) to store a single shared variable
B) to create a new thread
C) to store an array of values
D) to increase CPU performance

46. What does a multiprocessing.Array store?
A) a list of shared values
B) a single variable
C) a queue of tasks
D) a process lock

47. What is a Future in asyncio?
A) A completed coroutine
B) A coroutine scheduler
C) A placeholder for a result that will be available later
D) A thread manager

48. Which asyncio function is suitable for background coroutine execution?
A) asyncio.wait()
B) asyncio.create_task()
C) asyncio.timeout()
D) asyncio.defer()

49. Which is not a benefit of asyncio?
A) High thread overhead
B) Efficient task switching
C) Handles thousands of operations
D) Single-threaded

50. What is a race condition?
A) A task that runs faster than others
B) A condition where threads work cooperatively
C) Concurrent access to shared data without synchronization
D) Waiting for a lock indefinitely

51. What is the correct way to create a thread in Python?
A) thread = Threading.Process(target=function_name)
B) thread = threading.Thread(target=function_name)
C) thread = multiprocessing.Thread(target=function_name)
D) thread = process.Thread(target=function_name)

52. What does thread.join() do?
A) stops the thread
B) waits for the thread to finish execution
C) permanently blocks all other threads
D) restarts the thread

53. What are the main stages of a thread's lifecycle?
A) creation → running → execution → termination
B) running → execution → restart → termination
C) initialization → stop → kill → resume
D) loading → execution → pause → end

54. Why do we need thread synchronization?
A) to execute multiple threads in parallel
B) to prevent race conditions when accessing shared resources
C) to increase the number of threads
D) to reduce memory usage

55. What causes a deadlock?
A) Infinite loop
B) Incorrect output
C) Circular waiting on locks
D) Missing function

56. What role does message passing play in SPMD systems?
A) It eliminates the need for task synchronization
B) It facilitates communication between distributed memory nodes
C) It accelerates computation by bypassing processor coordination
D) It reduces the number of processors required

57. What is a major problem with CPU cycles in personal computers?
A) they are too slow
B) most CPU cycles are wasted
C) they cannot execute parallel tasks
D) they consume too much memory

58. What does a multi-core processor do?
A) reduces processor speed
B) acts as multiple CPUs in one
C) uses only one core at a time
D) prevents parallel execution

59. What is the effect of adding processors beyond the break-point?
A) speedup increases exponentially
B) speedup stays the same
C) speedup increases slightly
D) processing speed decreases

60. When is it beneficial to add more processors?
A) When below the break-point
B) When above the break-point
C) When running only serial tasks
D) When CPU usage is low

61. What is the purpose of inter-cluster connections in grid computing?
A) to improve data transfer between clusters
B) to slow down processing speed
C) to increase power consumption
D) to reduce the number of processors

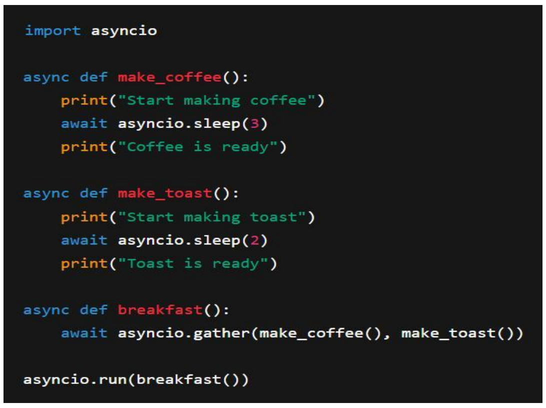
62. What is a livelock?
A) Threads are idle
B) Threads complete tasks early
C) Threads work but make no progress
D) Code crashes randomly

63. What happens if you await a Future?
A) It runs immediately
B) It throws an error
C) It waits until .set_result() is called
D) It restarts the event loop

64. What is the difference between a Task and a Future?
A) Tasks are synchronous
B) Tasks auto-schedule coroutines; Futures don't
C) Tasks are lower-level
D) Futures can't be awaited

65. What method manually sets a Future's result?
A) future.complete()
B) future.set_value()
C) future.set_result()
D) future.done()

66. What prints first in breakfast()?
A) Coffee is ready
B) Start making toast
C) Toast is ready
D) Start making coffee



```python
import asyncio

async def make_coffee():
    print("Start making coffee")
    await asyncio.sleep(3)
    print("Coffee is ready")

async def make_toast():
    print("Start making toast")
    await asyncio.sleep(2)
    print("Toast is ready")

async def breakfast():
    await asyncio.gather(make_coffee(), make_toast())

asyncio.run(breakfast())
```

67. Which is more efficient for managing many I/O tasks?
A) Threading
B) asyncio
C) multiprocessing
D) fork

68. Which function pauses a coroutine without blocking the event loop?
A) time.sleep()
B) asyncio.pause()
C) asyncio.sleep()
D) asyncio.stop()

69. Which synchronization technique allows only one thread to execute at a time?
A) semaphore
B) lock
C) event
D) threadpool

70. What is the main purpose of using a semaphore?
A) to limit the number of threads that can access a resource
B) to block thread execution
C) to allow unlimited access to threads
D) to execute all threads sequentially

71. How do you create a lock in Python?
A) lock = threading.Semaphore()
B) lock = threading.Lock()
C) lock = threading.ThreadLock()
D) lock = threading.Mutex()

72. What does executor.map() do?
A) applies a function to multiple arguments in parallel
B) creates a new process pool
C) starts only one process
D) terminates the executor

73. What does executor.submit() do in a process pool?
A) submits a function to be executed asynchronously
B) starts a new process
C) blocks execution until all tasks finish
D) terminates all processes

74. Which module provides ProcessPoolExecutor?
A) os
B) threading
C) concurrent.futures
D) multiprocessing

75. What are the benefits of a process pool?

A) reuses processes for multiple tasks
B) eliminates the need for CPUs
C) creates new CPU cores
D) makes multiprocessing unnecessary

76. In asyncio, what is a coroutine?

A) Threaded task
B) Class instance
C) Async function object
D) Event handler

77. When should you use asyncio.Future?

A) When you want automatic coroutine execution
B) For manual control of async results
C) For writing sync code
D) For running multiple tasks

78. Which of the following is a correct syntax?

A) await def greet():
B) def async greet():
C) async def greet():
D) coroutine greet():

79. What does thread starvation mean?

A) Threads consume too much CPU
B) Threads never complete
C) Some threads never get CPU time
D) Threads block on I/O

80. What's the best way to avoid deadlocks when using multiple locks?

A) Use a single lock only
B) Acquire locks randomly
C) Always acquire locks in the same order
D) Release locks immediately

81. What module is commonly used in Python for profiling performance?

A) profiler
B) asyncio
C) timeit
D) cProfile

82. What does ncalls mean in cProfile output?
A) CPU usage
B) Number of function calls
C) Network calls
D) None of the above

83. What is a process pool?
A) a collection of pre-initialized processes
B) a shared memory space
C) a queue for processes
D) a list of threads

84. What does lock.release() do?
A) unlocks the shared resource
B) starts a new process
C) stops all running processes
D) deletes the lock

85. What is tottime in cProfile?
A) Total runtime of program
B) Time in sub-calls
C) Time spent in the function itself
D) Unused time

86. What happens if multiple threads access shared data without synchronization?
A) efficiency improves
B) race conditions occur
C) threads get automatic priority
D) execution speeds up

87. What does with lock: do in threading?
A) acquires and releases the lock automatically
B) starts a new thread
C) terminates a thread
D) creates a new process

88. What is cumtime in profiling results?
A) Total cumulative time including sub-calls
B) Time without sub-calls
C) Time per line
D) Error time

89. Which of the following is a thread-safe data structure?
A) List
B) Dictionary
C) Queue
D) Set

90. What is the purpose of threading.Lock()?
A) Create threads
B) Delay execution
C) Synchronize access to shared resources
D) Improve speed

91. What's the issue if your program hangs during thread join()?
A) Logical error
B) Deadlock
C) Syntax error
D) Infinite recursion

92. Which of these is NOT a parallel programming bug?
A) Livelock
B) Starvation
C) Deadlock
D) Breakpoint

93. What makes a coroutine concurrent?
A) async/await
B) Threads
C) GIL
D) Main loop

94. Which will run first if both tasks are started at the same time?
A) One with shortest await duration
B) Random
C) Longest one
D) Last declared

95. What is an event in threading?
A) a function that permanently stops threads
B) a mechanism that makes threads wait until a condition is met
C) a way to create a new thread
D) a method to stop threads

96. Which function is used to set an event?

A) event.activate()

B) event.set()

C) event.wait()

D) event.run()

97. What does a multiprocessing.Lock do?

A) prevents multiple processes from accessing shared data at the same time

B) creates a new process

C) blocks all memory access

D) increases CPU performance

98. Why is synchronization needed in multiprocessing?

A) to prevent race conditions

B) to increase process execution time

C) to allow simultaneous memory access

D) to block all other processes

99. What are two shared memory objects in Python's multiprocessing?

A) value and array

B) queue and pipe

C) lock and semaphore

D) thread and event

100. What is shared memory in multiprocessing?

A) memory that all processes can access directly

B) memory assigned to one thread

C) a type of process pool

D) a way to store logs