----------1----------

Ktoré z nasledujúcich tvrdení o Java multithreadingu sú správne?
✅ Java poskytuje vstavanú podporu pre vlákna cez triedu java.lang.Thread a rozhranie Runnable.
✅ Metóda join() zabezpečuje, že hlavné vlákno čaká na dokončenie spustených vlákien.
✅ Synchronized bloky umožňujú ochranu kritických sekcií pred súčasným prístupom viacerých vlákien.
❌ Java umožňuje vytvárať vlákna len prostredníctvom dedenia triedy Thread.

Čo znamená skratka JDK?
✅ Java Development Kit
❌ Java Debug Kit
❌ Java Deployment Kit
❌ Java Data Kit

Ktoré z nasledujúcich tvrdení popisujú správny postup inštalácie a spustenia Java programu?
✅ Na inštaláciu je potrebné stiahnuť a nainštalovať JDK (napr. OpenJDK alebo Oracle JDK) a následne overiť verziu príkazmi java -version a javac -version.
✅ Zdrojové súbory sa kompilujú do bajtkódu pomocou príkazu javac NazovSuboru.java.
✅ Program sa spúšťa pomocou príkazu java NazovSuboru, čím JVM spustí bajtkód.
❌ Java program sa spúšťa ako natívny binárny kód priamo na procesore bez použitia JVM.

Ktoré z nasledujúcich synchronizačných mechanizmov sú bežne využívané v Jave pre prácu s vláknami?
✅ Synchronized bloky a metódy – zabezpečujú exkluzívny prístup do kritických sekcií.
✅ Metóda join() – umožňuje hlavnému vláknu čakať na dokončenie podriadených vlákien.
✅ Semaphore – riadi prístup k zdieľaným zdrojom pomocou povolení.
❌ Thread.sleep() – používa sa na oneskorenie vykonávania, ale neslúži primárne na synchronizáciu.

----------2----------

Aký príkaz v OpenMP zabezpečí paralelizáciu cyklu?
✅ #pragma omp parallel for
❌ #pragma omp for parallel
❌ #pragma omp multithreaded
❌ #pragma omp loop

Aké techniky synchronizácie sú podporované v OpenMP?
✅ Barrier
✅ Critical
❌ Semaphore
❌ Mutex

Čo sa stane, ak v OpenMP nepoužijeme #pragma omp atomic pri zdieľaných premenných?
❌ Program sa zrýchli bez vedľajších účinkov
✅ Môže dôjsť k nesprávnym výsledkom v dôsledku súbehu
❌ Program sa automaticky synchronizuje bez potreby ďalších zásahov
❌ OpenMP vygeneruje chybu a program sa nespustí

Čo znamená "race condition" v paralelnom programe?
✅ Keď viaceré vlákna súťažia o prístup k rovnakej premennej bez synchronizácie
❌ Keď sa paralelný program vykonáva príliš pomaly
❌ Keď vlákna vykonávajú kód v sekvenčnom poradí
❌ Keď program obsahuje nekonečný cyklus

----------3----------

Aké paradigmy python podporuje?
✅ Imperatívne
✅ Funkcionálne
✅ Objektovo orientované
❌ Programovanie na nízkej úrovni (príklad Assembler)
❌ Čisté Funkcionálne programovanie

Pri používaní Pythonu, prečo Global Interpreter Lock (GIL) vplýva na výkon multithreading-u?
✅ Bráni paralelnému behu vlákien a umožňuje, aby sa v jednom procese vykonávalo vždy len jedno vlákno.
❌ Zvyšuje výkonnosť viacvláknového spracovania tým, že umožňuje súčasné spustenie všetkých vlákien na viacerých jadrách.
❌ Úplne zakáže viacvláknovosť, čím znemožní používanie vlákien v jazyku Python.
❌ Vplýva len na programy viazané na I/O, ale programy viazané na CPU spúšťajú vlákna v skutočnom paralelizme.

Čo je lepšie použiť namiesto multithreadingu pre úlohy náročné na CPU?
✅ Multiprocesing

Ktoré z nasledujúcich tvrdení o jazyku Python sú pravdivé?
✅ Python podporuje objektovo orientované aj funkcionálne programovanie.
❌ Kód jazyka Python sa pred spustením kompiluje priamo do strojového kódu.
✅ Python používa na definovanie blokov kódu namiesto zátvoriek {} odsadenie.
✅ Python má automatickú správu pamäte pomocou zberu odpadu.
❌ Python nepodporuje dynamické písanie.

Ktoré z nasledujúcich tvrdení sú pravdivé o multithreadingu a multiprocesingu v Pythone?
❌ V jazyku Python je multithreading najvhodnejší pre úlohy viazané na procesor.
✅ Global Interpreter Lock (GIL) zabraňuje skutočnému paralelnému vykonávaniu vlákien v jazyku Python pre úlohy viazané na CPU.
✅ Multiprocesing v jazyku Python umožňuje spúšťanie viacerých procesov na viacerých jadrách CPU, čím sa obchádza GIL.
✅ Modul na spracovanie vlákien možno použiť pre úlohy viazané na CPU aj na I/O.
✅ Multithreading v jazyku Python zlepšuje výkon úloh viazaných na I/O tým, že umožňuje beh iných vlákien, kým jedno čaká na I/O.

----------4----------

Ktoré z nasledujúcich tvrdení o funkcii pthread_create() sú správne?
✅ Funkcia pthread_create() sa používa na vytvorenie nového vlákna v POSIX systémoch.
✅ Posledným parametrom je ukazovateľ na funkciu, ktorú bude nové vlákno vykonávať.
❌ pthread_create() vracia priamo identifikátor vlákna namiesto návratovej hodnoty.
❌ Funkcia pthread_create() vždy vyžaduje atribúty vlákna ako povinný parameter.
❌ Vlákno vytvorené pomocou pthread_create() sa automaticky ukončí po 1 sekunde nečinnosti.

Ktoré z nasledujúcich mechanizmov sa používajú na synchronizáciu vlákien v Pthread?
✅ Mutexy (pthread_mutex_t) umožňujú vzájomné vylúčenie prístupu k zdrojom.
✅ Bariéry (pthread_barrier_t) zabezpečujú synchronizáciu viacerých vlákien na jednom mieste v programe.
❌ pthread_join() zabezpečuje exkluzívny prístup k zdieľaným zdrojom.
❌ Funkcia pthread_yield() sa používa na synchronizáciu medzi vláknami.

Akým spôsobom môže vlákno ukončiť svoju činnosť v Pthread?
✅ Volaním pthread_exit(), čím vlákno ukončí svoju činnosť a môže odovzdať návratovú hodnotu.

✅ Použitím funkcie pthread_cancel() môže byť vlákno ukončené z iného vlákna.
❌ Ak hlavné vlákno ukončí svoju činnosť, všetky ostatné vlákna sú automaticky ukončené.
✅ pthread_detach() umožňuje vláknu uvoľniť svoje zdroje po ukončení bez nutnosti čakania na join.

Ako môžu vlákna bezpečne zdieľať dáta medzi sebou?
✅ Použitím globálnych premenných chránených mutexmi.
✅ Použitím podmienkových premenných na signalizáciu zmien v dátach medzi vláknami.
✅ Použitím vlákno-lokálnych premenných (Thread-Local Storage, TLS) na uchovávanie údajov špecifických pre dané vlákno.
❌ Použitím globálnych premenných bez akejkoľvek synchronizácie.

----------5----------

Ktoré z nasledujúcich možností sú správne pri vytváraní gorutín v Go?
(50%)
✅ Použitie kľúčového slova go pred volaním funkcie
(50%)
✅ Gorutiny bežia súčasne a môžu byť plánované na rôzne jadrá
(-50%)
❌ Gorutiny sú vždy viazané na jedno jadro procesora
(-50%)
❌ Gorutiny sú náhradou za vlákna operačného systému

Doplňte správne slovo: V Go sa na synchronizáciu a komunikáciu medzi gorutinami najčastejšie používajú _.
✅ kanály

Ktoré tvrdenia o sync.Mutex sú správne?
(50%)
✅ sync.Mutex sa používa na ochranu zdieľaných zdrojov pred súčasným prístupom viacerých gorutín
(50%)
✅ Lock() a Unlock() musia byť vždy správne spárované, inak môže dôjsť k deadlocku
(-50%)
❌ sync.Mutex umožňuje rekurzívne zamykanie rovnakou gorutinou
(-50%)
❌ Mutex v Go nie je potrebný, pretože všetky gorutiny vždy bežia sekvenčne

Čo sa stane s gorutinami, keď hlavná (main) gorutina v Go skončí?
(50%)
✅ Všetky bežiace gorutiny sú okamžite ukončené bez ohľadu na ich stav
(50%)
✅ Na zabránenie predčasného ukončenia hlavnej gorutiny je možné použiť sync.WaitGroup
(-50%)
❌ Gorutiny pokračujú v behu aj po ukončení hlavnej funkcie
(-50%)
❌ Go runtime automaticky počká, kým sa všetky gorutiny dokončia pred ukončením programu

----------6----------

Aký prístup má Rust k zabráneniu dátovým pretekom v multithreadingu?
❌ Garbage collection na bezpečné spravovanie pamäte (Garbage collection to manage memory safely)
✅ Vynucovanie ownership a borrowing pravidiel pri kompilácii (Enforcing ownership and borrowing rules at compile time)

❌ Uzamknutie každej premennej ako predvolené opatrenie proti konfliktom (Locking every variable by default to prevent access conflicts)

Ako môžu vlákna bezpečne komunikovať v Ruste?
❌ Použitím globálnych premenných na zdieľanie dát (Using global variables to share data)
✅ Použitím std::sync::mpsc kanálov (Using std::sync::mpsc channels)
❌ Zakázaním Rustových ownership pravidiel (Disabling Rust's ownership rules)

Ktoré z nasledujúcich mechanizmov zabezpečujú bezpečnosť vlákien v Ruste?
✅ Ownership a borrowing pravidlá (Ownership and borrowing rules)
✅ std::sync::Mutex
❌ Globálne premenné s mut static (Global variables with mut static)
✅ std::sync::mpsc kanály (std::sync::mpsc channels)

Ktoré z nasledujúcich tvrdení o std::sync::mpsc kanáloch sú pravdivé?
✅ Umožňuje bezpečnú komunikáciu medzi vláknami pomocou odovzdávania správ (It allows threads to communicate safely using message passing)
❌ Umožňuje obojsmernú komunikáciu ako predvolenú (It allows bidirectional communication by default)
✅ Sender (tx) môže byť klonovaný na podporu viacerých producentov (The sender (tx) can be cloned to allow multiple producers)
❌ Receiver (rx) môže byť klonovaný na podporu viacerých spotrebiteľov (The receiver (rx) can be cloned to allow multiple consumers)

----------7----------

Aké výhody prináša použitie SemaphoreSlim pri multithreadingu v C#?
✅ Obmedzuje počet súbežne bežiacich úloh, čím zabraňuje preťaženiu systému.
❌ Meria čas vykonania programu v milisekundách.
❌ Zabezpečuje, že úlohy sa vykonávajú v presnom poradí, v akom boli spustené.
❌ Umožňuje vykonanie iba jednej úlohy naraz.

O čom je knižnica Task v C#?
✅ Task umožňuje vytvoriť a spravovať viacvláknové aplikácie pomocou asynchrónnych metód.
❌ Task je knižnica na správu pamäte a optimalizáciu využitia systémových zdrojov.
❌ Task poskytuje nástroje na prácu so súborovými operáciami a vstupno-výstupnými zariadeniami.
❌ Task slúži na implementáciu a správu grafických užívateľských rozhraní v C [aplikáciách.]

Ktoré z nasledujúcich tvrdení najlepšie popisuje správanie metódy Parallel.For v C#?
❌ Vykonáva všetky iterácie postupne v jednom vlákne.
✅ Vykonáva iterácie paralelne bez záruky poradia ich vykonávania.
❌ Zaručuje vykonávanie iterácií striktne v opačnom poradí.
❌ Používa sa na synchronné vykonávanie asynchrónnych operácií.

Ktoré z nasledujúcich kolekcií v C# sú vlákno-bezpečné a špeciálne navrhnuté pre použitie v prostredí s viacerými vláknami?
✅ ConcurrentBag.
✅ BlockingCollection.
❌ Dictionary.
❌ List.
❌ HashSet.
✅ ConcurrentDictionary.
❌ Observable.