TEMATICĂ DE CONCURS

pentru ocuparea postului vacant de Asistent universitar- perioada determinată

Postul 40 din Statul de functii al Departamentului de Inginerie Electrică și Tehnologia Informației

I. PROGRAMARE ORIENTATĂ OBIECT / LIMBAJE ORIENTATE OBIECT

1. Încapsularea și protejarea stării obiectelor

1.1. Ascunderea informației (*Information Hiding*) și niveluri de vizibilitate: utilizarea modificatorilor de acces și impactul lor asupra vizibilității membrilor la nivel de clasă, pachet și ierarhie de derivare. (*POO-1 Capitolul 1 – Secțiunea „Encapsulation and Data Hiding", Capitolul 2 – Secțiunea „Knowing the Difference Between the Interface and the Implementation"; POO-2 Capitolul 4 – Objects and Classes*)

1.2. Managementul accesului la starea internă: implementarea metodelor de acces (accesori și mutatori / proprietăți) pentru validarea datelor și asigurarea consistenței stării obiectului. (*POO-1 Capitolul 4 – Secțiunea „Accessors"; POO-2 4.2.3 – Mutator and Accessor Methods; POO-3 Capitolul 4 – Manipulating Objects, Capitolul 6 – Retrieving Information*).

1.3. Ciclul de viață al obiectelor: mecanisme de inițializare; rolul și tipologia constructorilor (impliciți, de inițializare, de copiere). (*POO-1 Capitolul 3 – Secțiunea „Constructors", Capitolul 4 – Secțiunea „Constructors"; POO-2 4.6 – Object Construction*)

1.4. Invarianții clasei: conceptul de invarianți și rolul constructorilor/metodelor în menținerea unui stat valid al instanței pe toată durata execuției. (*POO-3 Capitolul 3 – Creating other objects, Capitolul 4 – Manipulating objects*)

1.5. Managementul memoriei în copierea obiectelor: diferența la nivel de referință și alocare de memorie între copierea superficială (*Shallow Copy*) și copierea în profunzime (*Deep Copy / Clonare*). (*POO-1 Capitolul 3 – Secțiunea „Object Operations"; POO-2 6.1.9 – Object Cloning*).

2. Moștenire și ierarhii de clase

2.1. Relații între clase: diferența conceptuală și arhitecturală dintre moștenire (relația de subtipizare / derivare) și asocieri (relațiile de agregare și compoziție). (*POO-1 Capitolul 1 – Secțiunile „Inheritance" și „Composition", Capitolul 7 – Secțiunile „Inheritance" și „Composition"; POO-2 5.1 Classes, Superclasses, and Subclasses*).

2.2. Mecanismul de derivare și lanțul de inițializare: ordinea de apel a constructorilor în ierarhiile de clase. Transmiterea parametrilor către constructorii claselor de bază. (*POO-2 5.1.3 Subclass Constructors*)

2.3. Niveluri de abstractizare: rolul claselor abstracte versus rolul interfețelor. Contracte de implementare și limitări privind instanțierea și starea (atributele). (*POO-1 Capitolul 8 – Secțiunea „What Is a Contract?"; POO-2 5.6 – Abstract Classes, 6.1 – Interfaces*).

2.4. Suprascrierea și ascunderea membrilor: mecanismul de suprascriere a metodelor (*Method Overriding*) în clasele derivate comparativ cu ascunderea numelor (*Name Hiding / Shadowing*) pentru atribute și metode statice. (*POO-2 5.1.2 – Overriding Methods*).

2.5. Conversii de tip în ierarhii (*Type Casting*): reguli și siguranța conversiilor ascendente (*Upcasting*) și a conversiilor descendente (*Downcasting*). (*POO-2 5.1.8 – Casting*).

3. Polimorfism și legare dinamică
   3.1. Polimorfismul la compilare (*Static / Early Binding*): mecanismul de supraîncărcare a metodelor și constructorilor (*Method Overloading*) pe baza semnăturii (tipul, ordinea și numărul parametrilor). (*POO-1 Capitolul 3 – „Secțiunea Constructors", Capitolul 3 – Secțiunea „Operator Overloading"; POO-2 4.6.1 – Overloading*).
   3.2. Polimorfismul la execuție (*Dynamic / Late Binding*): mecanismul prin care mediul de execuție determină implementarea metodei ce trebuie apelată în funcție de tipul real al instanței (la runtime), și nu de tipul referinței. (*POO-1 Capitolul 1 – Secțiunea „Polymorphism"; POO-2 5.1.5 – Polymorphism, 5.1.6 – Understanding Method Calls*).
   3.3. Metode virtuale și abstracte: constrângeri și reguli de invocare a comportamentelor polimorfice. Diferența dintre metodele cu implementare implicită (virtuale) și cele care impun doar un contract (abstracte). (*POO-2 5.6 – Abstract Classes, 5.1.6 – Understanding Method Calls*).
   3.4. Structuri de date eterogene: utilizarea polimorfismului pentru gruparea, iterarea și prelucrarea uniformă a instanțelor de tipuri derivate diferite prin intermediul referințelor de tip clasă de bază sau interfață. (*POO-2 5.3 – Generic Array Lists*)
   3.5. Identificarea tipurilor la rulare (*RTTI - Run-Time Type Information*): operatori pentru testarea și identificarea tipului exact al unui obiect la execuție. Utilitate, limitări și impactul asupra designului orientat pe obiecte. (*POO-2 5.1.8 – Casting, 5.1.9 Pattern Matching for instanceof, 5.9 – Reflection*)

II. SISTEME DE OPERARE

4. Sincronizarea proceselor și Interblocarea (*Process Synchronization & Deadlocks*)
   - 4.1. Problema secțiunii critice: definirea formală și cerințele fundamentale pentru o soluție validă (excludere mutuală, progres, așteptare mărginită). (*SO-1 5.1 – Principles of Concurrency, 5.3 – Mutual Exclusion: Hardware Support; SO-2 6.2 – The Critical-Section Problem*).
   - 4.2. Mecanisme de sincronizare hardware și software: suportul arhitectural pentru sincronizare (instrucțiuni atomice precum *Test-and-Set*, *Compare-and-Swap*). Semafoare (de contorizare și binare). Mutex-uri (lacăte de excludere mutuală). Monitoare: variabile de condiție și abstractizarea sincronizării. (*SO-1 5.4 – Semaphores, 5.5 (Monitors); SO-2 6.4 – Synchronization Hardware, 6.5 – Semaphores, 6.7 – Monitors*).
   - 4.3. Probleme clasice de sincronizare: modelarea și rezolvarea problemelor tipice de concurență (ex: problema producător-consumator, problema cititorilor și scriitorilor, problema filozofilor). (*SO-1 5.7 – Readers/Writers Problem; SO-2 6.6 – Classic Problems of Synchronization*).
   - 4.4. Interblocarea (*Deadlock*). Condițiile necesare și suficiente pentru apariția interblocării. Mecanisme de abordare: prevenire, evitare (algoritmul bancherului), detectare și recuperare din starea de interblocare. (*SO-1 6.1 – Principles of Deadlock, 6.2 – Deadlock Preventioan, 6.3 – Deadlock Avoidance, 6.4 – Deadlock Detection; SO-2 6.9 – Deadlocks*).
5. Gestiunea memoriei și memoria virtuală (*Memory Management & Virtual Memory*)
   - 5.1. Spațiul de adrese și alocarea fizică: spațiul de adrese logic (virtual) vs. fizic. Unitatea de management a memoriei (MMU). Alocarea contiguă a memoriei. Fragmentarea internă și externă (*SO-1 7.1 – Memory Management Requirements, 7.2 – Memory Partitioning; SO-2 7.1 – Main Memory / Background, 7.3 – Contigous Memory Allocation*).
   - 5.2. Paginare și segmentare (*Paging & Segmentation*): arhitectura paginării – structura hardware, tabele de pagini și rolul memoriei asociative (*TLB - Translation Look-aside Buffer*); arhitectura tabelelor de pagini (ierarhice / multinivel, tabele inversate). Segmentarea memoriei și maparea logică a programelor (*SO-1 7.3 – Paging, 7.4 – Segmentation; SO-2 7.4 – Paging, 7.5 – Structure of the Page Table, 7.6 – Segmentation*).
   - 5.3. Memoria virtuală (*Virtual Memory*): paginarea la cerere (*Demand Paging*) și mecanismul de tratare a defectelor de pagină (*Page Faults*). (*SO-1 8.1 – Hardware and Control Structures; SO-2 8.1 – Virtual Memory/Background, 8.2 – Demand Paging*).
   - 5.4. Algoritmi de înlocuire a paginilor: analiza comparativă a algoritmilor FIFO (*First-In-First-Out*), OPT (*Optimal*), LRU (*Least Recently Used*) și a algoritmilor bazați pe aproximare (*Second-Chance / Clock*). (*SO-1 8.2 – Operating System Software; SO-2 8.4 – Page Replacement*).
6. Planificarea pceselor roși a firelor de execuție (*Process & Thread Scheduling*)

6.1. Concepte fundamentale de planificare: ciclul procesor-intrare/ieșire (*CPU-I/O Burst Cycle*); criterii de planificare (gradul de utilizare a procesorului, debitul / throughput, timpul de răspuns, timpul de așteptare, timpul de rotație / turnaround time); planificare preemptivă vs. planificare nepreemptivă. (*SO-1 9.1 – Types of Processor Scheduling; SO-2 5.1 – Basic Concepts, 5.2 – Scheduling Criteria*).

6.2. Algoritmi clasici de planificare a procesorului: modul de funcționare și analiza algoritmilor – *First-Come, First-Served* (FCFS), *Shortest-Job-First* (SJF), *Shortest-Remaining-Time-First* (SRTF), *Round-Robin* (RR), planificarea bazată pe priorități. (*SO-1 9.2 – Scheduling Algorithms; SO-2 5.3 – Scheduling Algorithms*).

6.3. Arhitecturi avansate de cozi: cozi multinivel (*Multilevel Queue Scheduling*). cozi multinivel cu feedback (*Multilevel Feedback Queue Scheduling*); mecanisme de ajustare dinamică a priorităților și prevenirea înfometării (*Starvation / Aging*). (*SO-1 9.2 – Scheduling Algorithms; SO-2 5.3 – Scheduling Algorithms*).

6.4. Planificarea în sisteme multiprocesor (SMP): partajarea sarcinii (*Load Balancing*), afinitatea procesorului (*Processor Affinity*) și alocarea firelor de execuție pe nuclee multiple (*SO-1 10.1 – Multiprocessor Scheduling; 5.5 – Multiple-Processor Scheduling*).

III. REȚELE DE CALCULATOARE

7. Performanța rețelelor și mecanisme de control (*Network Performance & Control*)

7.1. Metrici fundamentale de performanță: componentele întârzierii (*Delay*) – procesare nodală, așteptare în cozi (*queueing delay*), transmisie și propagare. Lățimea de bandă (*Bandwidth*), debitul util (*Throughput*) și conceptul critic de produs Lățime de Bandă X Întârziere (*Bandwidth-Delay Product - BDP*) în dimensionarea bufferelor. (*RC-1 Capitolul 1 (Computer Networks and the Internet) – Secțiunea 1.4 (Delay, Loss, and Throughput in Packet-Switched Networks); RC-2 1.5 – Performance*).

7.2. Controlul erorilor (*Error Control*): mecanisme de detecție a erorilor (*Checksums*, *Cyclic Redundancy Check - CRC*). Protocoale de tip ARQ (*Automatic Repeat reQuest*) – analiza comparativă între *Stop-and-Wait*, *Go-Back-N* și *Selective Repeat*. (*RC-1 Capitolul 6 – Secțiunea 6.2 (Error-Detection and Correction Techniques), Capitolul 3 – Secțiunea 3.4 (Principles of Reliable Data Transfer); RC-2 2.4 – Error Detection, 2.5 – Reliable Transmission*).

7.3. Controlul fluxului (*Flow Control*): protejarea receptorului; mecanisme bazate pe fereastră glisantă (*Sliding Window*) la nivelul legătură de date și la nivelul transport. (*RC-1 Capitolul 3 – Secțiunea 3.5 (Connection-Oriented Transport: TCP); RC-2 2.5 – Reliable Transmission, 5.2 – Reliable Byte Stream (TCP)*).

7.4. Controlul Congestiei (*Congestion Control*): diferența conceptuală dintre controlul fluxului (*end-to-end*) și controlul congestiei (protejarea rețelei). Mecanisme specifice TCP – creștere aditivă / descreștere multiplicativă (AIMD), *Slow Start*, *Fast Retransmit* și *Fast Recovery*. (RC-1 Capitolul 3 – Secțiunea 3.6 (Principles of Congestion Control), Capitolul 3 – Secțiunea 3.7 (TCP Congestion Control); RC-2 6.3 – TCP Congestion Control).

8. Nivelul legătură de date: comutația (*switching*) în rețele locale (*Switched Ethernet*)

8.1. Topologii și izolarea traficului: diferența de arhitectură și performanță între rețelele cu mediu partajat (CSMA/CD / Ethernet semi-duplex) și Ethernet-ul comutat full-duplex. Delimitarea domeniilor de coliziune și a domeniilor de difuzare (*Broadcast Domains*). (*RC-1 Capitolul 6 – Secțiunea 6.3 (Multiple Access Links and Protocols), Capitolul 6 – Secțiunea 6.4 (Switched Local Area Networks; RC-2 2.6 – Multiaccess Networks, 3.2 – Switched Ethernet*);

8.2. Arhitectura comutatoarelor (*Switches*): mecanismul de auto-învățare (*Self-learning*) și filtrare/retransmitere a cadrelor. Popularea dinamică a tabelelor de comutație (*Forwarding / MAC Tables*). (*RC-1 Capitolul 6 – Secțiunea 6.4 (Switched Local Area Networks); RC-2 3.2 – Switched Ethernet*).

8.3. Redundanță și prevenirea buclelor la nivel 2: problema furtunilor de broadcast (*Broadcast Storms*). Algoritmul arborelui de acoperire (*Spanning Tree Protocol - STP*) și principiile de funcționare ale variantelor rapide (RSTP). (*RC-2 3.2 – Switched Ethernet*)

8.4. Virtualizarea rețelelor locale (VLANs): arhitectura rețelelor virtuale locale, rutarea inter-VLAN și mecanismele de etichetare a cadrelor pentru legăturile trunchi (*VLAN Trunking / IEEE 802.1Q*). (*RC-1 Capitolul 6 – Secțiunea 6.4 (Switched Local Area Networks); RC-2 3.2 – Switched Ethernet*).

9. Algoritmi și protocoale de rutare (*Routing Algorithms & Protocols*)

9.1. Paradigme fundamentale de rutare: algoritmi bazați pe starea legăturilor (*Link-State*) – funcționare pe baza algoritmului lui Dijkstra. Algoritmi bazați pe vectori de distanță (*Distance-Vector*) – funcționare pe baza ecuațiilor Bellman-Ford. (*RC-1 Capitolul 5 – Secțiunea 5.2 (Routing Algorithms); RC-2 3.4 – Routing*).

9.2. Probleme clasice în rutarea distribuită: fenomenul numărării la infinit (*Count-to-Infinity*) în algoritmii vectorilor de distanță și mecanisme de soluționare (ex. *Poisoned Reverse*). (*RC-1 Capitolul 5 –Secțiunea 5.2 (Routing Algorithms); RC-2 3.4 – Routing*).

9.3. Arhitectura ierarhică a rutării (Sisteme Autonome - AS): nevoia de ierarhizare (scalabilitate și autonomie administrativă). Rutarea interioară (Intra-AS) – concepte din OSPF (*Link-State*) și RIP (*Distance-Vector*). (*RC-1 Capitolul 5 – Secțiunea 5.3 (Intra-AS Routing in the Internet: OSPF), Capitolul 5 – Secțiunea 5.4 (Routing Among the ISPs: BGP); RC-2 4.1 – Global Internet*).

9.4. Rutarea bazată pe politici (*Inter-AS / Policy Routing*): concepte fundamentale ale protocolului BGP (*Border Gateway Protocol*) și paradigma de rutare bazată pe vectori de cale (*Path-Vector*).

IV. TEHNICI AVANSATE DE PROGRAMARE

10. Modelul obiectual, ciclul de viață și managementul resurselor (clase și obiecte)

10.1. Abstracția datelor și modelul obiectual: trecerea de la tipuri de date fundamentale la tipuri definite de utilizator. Maparea conceptelor din domeniul problemei în starea și comportamentul entităților software. (*TAP-1 Capitolul 16 (Classes) – Secțiunile 16.1 (Introduction) și 16.2 (Class Basics*)).

10.2. Ciclul de viață al obiectelor și alocarea memoriei: mecanisme de inițializare și distrugere. Diferențele arhitecturale, de performanță și de vizibilitate între alocarea statică, alocarea automată (pe stivă / stack) și alocarea dinamică (pe movilă / heap).

(*TAP-1 Capitolul 11 (Select Operations) – Secțiunea 11.2 (Free Store), Capitolul 17 (Construction, Cleanup, Copy, and Move) – Secțiunea 17.2 (Constructors and Destructors*).

10.3. Managementul resurselor și siguranța memoriei: paradigma RAII (*Resource Acquisition Is Initialization*) ca tehnică avansată de prevenire a scurgerilor de memorie (*memory leaks*) prin legarea ciclului de viață al unei resurse de ciclul de viață al unui obiect local. (*TAP-1 Capitolul 13 (Exception Handling) – Secțiunea 13.3 (Resource Management); TAP-2 Capitolul 4 (Smart Pointers)*).

10.4. Semantica de copiere versus Semantica de mutare: diferența teoretică între copierea stării unui obiect (cu implicații asupra performanței) și transferul de proprietate asupra resurselor (*Move Semantics*), esențială în programarea de înaltă performanță. (*TAP-1 Capitolul 17 (Construction, Cleanup, Copy, and Move) – Secțiunea 17.5 (Copy and Move); TAP-2 Capitolul 5 (Rvalue References, Move Semantics, and Perfect Forwarding) – Item 23 (Understand std::move and std::forward*).

11. Polimorfism ad-hoc și extensibilitate semantică (supraîncărcarea operatorilor)

11.1. Fundamentele polimorfismului ad-hoc: conceptul de supraîncărcare a operatorilor ca funcții speciale. (*TAP-1 Capitolul 18 (Operator Overloading) - Secțiunea 18.1 (Introduction), Capitolul 18 (Operator Overloading) - Secțiunea și 18.2 (Operator Functions*).

11.2. Operatori cu semantică specială: gestionarea atribuirii (operatorul =) și prevenirea auto-atribuirii (self-assignment). Supraîncărcarea operatorilor de dereferențiere și acces la membri pentru crearea de pointeri inteligenți (*Smart Pointers*). (*TAP-1 Capitolul 17 – Secțiunea 17.5.3 (Copy Assignment), Capitolul 19 (Special Operators) – Secțiunea 19.2.2 (Dereferencing); TAP-3 Capitolul 5 (Utilities), secțiunea 5.2 (Smart Pointers)*).

11.3. Obiecte apelabile (functori – *function objects*): supraîncărcarea operatorului de apel de funcție (). Conceptul de "obiecte care se comportă ca funcții" și rolul lor ca fundament teoretic pentru funcțiile anonime (expresii Lambda). (*TAP-1 Capitolul 19 (Special Operators) – Secțiunea 19.2.2 (Function Call), Capitolul 11 (Select Operations) – Secțiunea 11.4 (Lambda Expressions); TAP-3 Capitolul 6 (The Standard Template Library) – Secțiunea 6.10 (Function Objects), Capitolul 3 (New Language Features) – Secțiunea 3.1.10 (Lambdas)*).

12. Programare generică, containere

12.1. Paradigma programării generice: crearea de componente software reutilizabile și independente de tipul datelor. Structura tripartită a bibliotecilor standard moderne: containere, iteratori și algoritmi. (*TAP-3 Capitolul 6 (The Standard Template Library) – Secțiunile 6.1 (STL Components), 6.2 (Containers), 6.3 (Iterators) și 6.4 (Algorithms)*).

12.2. Taxonomia containerelor: containere secvențiale: liste înlănțuite, tablouri dinamice, cozi cu două capete (*deque – Double-Ended Queue*). Containere asociative: structuri bazate pe arbori echilibrați (hărți/dicționare ordonate) și structuri bazate pe tabele de dispersie (*Hash Tables*). Adaptoare de containere: abstractizarea structurilor de bază pentru a crea stive (LIFO) și cozi (FIFO). (*TAP-3 Capitolul 7 (STL Containers) – Secțiunile 7.1 (Common Container Abilities and Operations), 7.2 (Arrays), 7.3 (Vectors),*

7.4 (Deques), 7.5 (Lists), 7.6 (Forward Lists), 7.7 (Sets and Multisets), 7.8 (Map and Multimaps), 7.9 (unordered Containers), Capitolul 12 (Special Containers) – Secțiunile 12.1 (Stacks) și 12.2 (Queues))

12.3. Abstracția iteratorului: conceptul de pointer generalizat. Modul în care iteratorii acționează ca o interfață (punte de legătură) între structura internă, opacă a unui container și algoritmii care prelucrează datele. (*TAP-3 Capitolul 9 (STL Iterators) – Secțiunile 9.1 (Header Files for Iterators) și 9.2 (Iterator Categories)*).

Bibliografie:

- (**POO-1**) Matt Weisfeld, "The Object-Oriented Thought Process", 5th Edition, Addison-Wesley Professional, 2019.
  http://80.96.11.104/P40/POO-1.pdf
- (**POO-2**) Cay S. Horstmann, "Core Java, Volume I: Fundamentals", 13th Edition, Pearson, 2025.
  http://80.96.11.104/P40/POO-2.pdf
- (**POO-3**) Matthias Noback, "Object Design Style Guide", 1st Edition, Manning, 2019.
  http://80.96.11.104/P40/POO-3.pdf
- (**SO-1**) A. Silberschatz, P. B. Galvin, G. Gagne "Operating System Concept", 10th Edition, Wiley, 2018.
  http://80.96.11.104/P40/SO-1.pdf
- (**SO-2**) W. Stallings "Operating Systems: Internals and Design Principles", 9th Edition, Pearson, 2017.
  http://80.96.11.104/P40/SO-2.pdf
- (**RC-1**) J. F. Kurose, K. W. Ross, "Computer Networking: A Top-Down Approach", 8th Edition, Pearson, 2020.
  http://80.96.11.104/P40/RC-1.pdf
- (**RC-2**) L. L. Peterson, B. S. Davie, "Computer Networks: A Systems Approach", 6th Edition, Morgan Kaufmann, 2021.
  http://80.96.11.104/P40/RC-2.pdf
- (**TAP-1**) Bjarne Stroustrup, "The C++ Programming Language", Addison-Wesley Professional, 2013.
  http://80.96.11.104/P40/TAP-1.pdf
- (**TAP-2**) Scott Meyers, "Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14", 1st Edition, O'Reilly Media, 2014.
  http://80.96.11.104/P40/TAP-2.pdf
- (**TAP-3**) Nicolai M. Josuttis, "The C++ Standard Library: A Tutorial and Reference", 2nd Edition, Addison-Wesley Professional, 2012.
  http://80.96.11.104/P40/TAP-3.pdf