1. **tétel**

***A C programozási nyelv I.:*** *Adattípusok, deklarációik, feltételes utasítások.****Általános ismeretek:*** *adat és információ, entrópia fajtái, kifejezések infix és postfix alakja.****Mesterséges intelligencia:*** *Keresési problémák állapottér-reprezentációja, példák.  
Neminformált keresési eljárások (mélységi, szélességi, optimális).*

**Elemi adattípusok:** int, char, float, double  
**Összetett adattípusok:** tömb, struktúra  
**pointer típus:** a memóriacímet tartalmazza

Deklaráláskor a rendszer lefoglalja a típusnak megfelelő memória területet. A const nem változik.

**if-else:** elágazás  
**switch-case:** többfelé történő elágazás

**Adat:** Olyan jelsorozat, amely alkalmas a feldolgozásra  
**Információ:** Értelmezett adat  
**Entrópia:** Egy jelsorozat információtartalmát fejezi ki.

**Minimális entrópia:** mindig ugyanaz történik: 0  
**Maximális entrópia:** ha a bekövetkezési esélyek azonosak: log2n  
 **Relatív entrópia:** A tényleges entrópia és a maximális hányadosa

**infix:** Az operátorokat az operandusok közé tesszük  
**postfix:** Az operátorokat az operandusok után használjuk

**Állapottér reprezentáció:** Egy probléma megoldásához szükséges tér, tulajdonságok és jellemzők modellezése. Tartalmazza az összes lehetséges állapotot. A megoldáshoz a **kezdőállapotból** valamelyik **célállapotba** kell eljutni. Az állapotok változását **operátorokkal (műveletekkel)** érjük el.  
**Költség:** az operátor használatának költsége  
**Előfeltételek:** milyen állapotban használható az adott operátor

**Nem informált keresési eljárások:** Nincs információ az állapotokról.

**Szélességi keresés:** Minden csomópontot megvizsgál, mielőtt a következő szintre lépne. FIFO (sor). Idő- és tárigénye nagy. Teljes és ha minden műveletnek ugyanannyi a költsége, akkor optimális is.

**Mélységi keresés:** A keresési fa mélyebb csomópontjai felé halad. LIFO (verem). Időigénye nagy. Tárigény gráf esetén annyi, mint az időigény. Fakeresésnél lineáris tár. Ha az állapottér végtelen, akkor nem teljes. Nem optimális.

**Optimális kereső:** Minden csúcshoz tárolja az út költségét. Mindig a kisebb költségű lépést választja. Teljes és optimális.

1. **tétel**

***A C programozási nyelv II.****: Ciklusszervezési lehetőségek, függvénykezelés, paraméterkiértékelés, hatáskörkezelés (statikus, dinamikus).****Általános ismeretek:*** *számrendszerek, számábrázolás (fix és lebegőpontos), karakter, szöveg és logikai adat ábrázolása.****Mesterséges intelligencia:*** *A heurisztika fogalma, példák. A\* algoritmus. Az A\* algoritmus teljessége. Kétszemélyes, teljes információjú, determinisztikus játékok: a stratégia fogalma, minimax-algoritmus, alfa-béta vágás.*

Elő és utótesztelő ciklusok.  
**for:** inicializáció és feltétel, majd a törzs végrehajtása után léptetés  
**while:** nincs init csak feltétel, a léptetés a törzsben van  
**do-while:** egyszer mindenképpen végre lesz hajtva a törzs

**függvények:** többször meghívható alprogramok. A visszatérési értéküket a legelején kell megadni. A paraméterek lokálisak lesznek. A paraméterek átadása érték szerint történik.

**lokális változó:** csak a blokkon belül érhető el  
**globális változó:** a függvényeken kívül vannak deklarálva, ezért mindenhol elérhetőek

**statikus:** csak egyszer jönnek létre és egyszer kapnak értéket

**fix pontos ábrázolás:** egész számokhoz (kettes komplemens)  
**lebegőpontos ábrázolás:** törtekhez (előjel, mantissza, kettedes eltolás)  
**karakter ábrázolás**: kódtáblák (ASCII, Unicode)  
**szöveg ábrázolása:** karakter tömbben, kell egy \0 zárókarakter is  
**logikai érték ábrázolása:** 1byte-on, 1 vagy 0

**Heurisztika:** Egy állapot költségének becslése. Nem becsüli túl a költséget. A célállapot értéke 0.

**A\* algoritmus:** A legolcsóbb utat keresi. A megtett út költsége és a célba jutás költségének összege. Teljes és optimális.

A **kétszemélyes játékok** kompetitív multiágens környezetek.  
**Teljes információjú játék:** Minden eleméről információnk van.  
**Determinisztikus:** Nincs szerepe a véletlennek.

**Stratégia:** Minden játékálláshoz meghatározza a lépéseket.  
**Nyerő stratégia:** Az ellenfél lépéseitől függetlenül győzelemre vezet  
**Nem-vesztő stratégia:** Az ellenfél lépéseitől függetlenül, nem vezet vereséghez.

**minimax:** A döntést a minimum és maximum keresés eredményéből hozza meg.  
**alfa-béta:** Az eredménye ugyanaz, mint a minimax-nak, de a szükségtelen ágakat nem járja be. Az alfa maximumot, a béta minimumot keres.

1. **tétel**

***A C programozási nyelv III.:*** *Tömbök, mutatók, dinamikus memóriakeresés. Karakterlánckezelés****Formális nyelvek****: Környezetfüggetlen grammatikák, CNF, CYK algoritmus.****Hálózatok****: adatkapcsolati protokollok, rétegek. Lokális hálózatok. Az internet alapjai, HTML.*

**tömb:** azonos típusú adatokat tárol. A mérete nem dinamikus.  
**mutató:** memóriacímet tartalmaz. \*-gal jelöljük, a címet &-el (címképző) kapjuk meg. A mutatóban szereplő címhez tartozó változót a \*-gal (indirekció) kapjuk meg  
  
**dinamikus memóriakezelés:** A foglalást és a felszabadítást mi kezeljük. malloc() foglal a free() felszabadít. A malloc() paramétere int, és annyi bájtot foglal. A free() paramétere a felszabadítani kívánt pointer. calloc() első paramétere a darabszám a második a méret.  
  
**Karakterlánckezelés:** Karaktertömbben tároljuk. Fix vagy változó hosszúságú lehet. \0 karakter zárja le, ezért a hossza mindig egyel nagyobb.  
Műveletek: scanf, gets, printf, puts, strcpy, strcat, strlen, strcmp

**Környezetfüggetlen nyelvtan:** Ha a bal oldalon nemteminális van és a jobb oldalán nemterminálisokból és terminálisokból álló szó van. Szintaxis meghatározása.  
Minden környezetfüggetlen nyelvhez létezik olyan vele ekvivalens környezetfüggetlen nyelv, ami környezetfüggő is.

**Chomsky-féle normálalak (CNF):** Ha egy környezetfüggetlen nyelvtan minden szabálya A->a vagy A->BC alakú, ahol A, B, C eleme N, és a eleme T. Tetszőleges lambdamentes nyelvet generáló környezetfüggetlen nyelvtanhoz létezik vele ekvivalens Chomsky-féle normálalak.

**CYK:** vizsgálja, hogy egy szó eleme-e a nyelvnek.

**Adatkapcsolati protokollok:** Feladatuk az összeállított keret átvitele két csomópont között.  
**Elemi protokollok:** szimplexek; korlátozás nélküli, megáll és vár, zajos csatornálhoz  
PPP (internetkapcsolat), ALOHA (véletlen hozzáférés), CSMA (figyeli mikor szabad a csatorna), HDLC (ellenőrző összeget használ)  
Adatátviteli módok: szimplex, fél-duplex, duplex  
**Adatkapcsolati alrétegek: MAC:** a csatornához való hozzáférést vezérli, **LLC:** hibajavítás

**LAN:** helyi hálózat…blabla  
**Topológiák:** gyűrű, sin, csillag, teljes, fa

**Internet:** Decentralizált globális hálózat. Szolgáltatások: WWW, mail, telnet, SSH, FTP  
IP alapú kommunikáció.

**HTML:** leíró nyelv, strukturált szerkezet. A tageknek a szemantikai jelentése is fontos a HTML5 óta.

1. **tétel**

***Adatszerkezetek és algoritmusok****: Funkcionális specifikáció. Programozási tételek: keresés, rendezés, döntés, kiválogatás.****Adatbázis-rendszerek****: A relációs adatmodell. Egyed, attribútum, reláció és kapcsolat. Kulcs, idegen kulcs, hivatkozási integritás. Kényszerfeltételek az adatbázis elemein. Triggerek.****Számításelmélet****: Turing-gépek, Church-tézis, megállási probléma, algoritmikusan eldönthetetlen problémák. Logikai függvények megadása, KNF, DNF, logikai hálózatok.*

**Funkcionális specifikáció:** A felhasználó szemszögéből írja le az igényeket. Nem határozza meg a rendszer belső működését.

**Eldöntés tétele:** Egy tulajdonságról eldönti, hogy van-e a sorozatban olyan elem, amelyre igaz.  
**Lineáris keresés tétele:** Ugyanaz, mint az eldöntés, csak megadja az elem sorszámát is.  
**Bináris keresés:** Rendezett sorozaton. A középső elemmel összehasonlítva a keresendőt, szűkíti a tartományt, attól függően, hogy kisebb vagy nagyobb.  
**Kiválogatás tétele:** Egy adott tulajdonságú elemek sorszámát gyűjti egy másik tömbbe.  
**Buborék rendezés:** Minden körben a szomszédos elemeket rendezi sorba, így az utolsó elem a legnagyobb lesz. Minden kör eggyel rövidebb, mert a legnagyobbat már nem kell rendezni.

**Relációs adatmodell:** A reláció a domainek Descartes-szorzatának részhalmaza.  
**Egyed:** rekord, egy relációban minden előfordulás egy egyed  
**Attribútum:** az egyed tulajdonságai  
**Kapcsolat:** az egyedek közötti viszony  
**Kapcsolat típusok:** egy-egy, egy-több, több-több  
**Kulcs:** Az egyed azonosítására használt tulajdonság. Az elsődleges kulcs egyértelműen meghatároz egy egyedet. Az idegen kulcs egy olyan tulajdonság, ami egy másik relációban kulcs.  
**Hivatkozási integritás:** Nem lehet olyan idegen kulcs, amely nem egyezik meg a hivatkozott reláció valamelyik elsődleges kulcsával.  
**Kényszerfeltételek:** Nem NULL, értékkészlet, egyedi kulcs  
**Trigger:** Egy eseményre, a feltételtől függően végrehajt egy utasítást  
  
**Turing-gép:** Olyan véges automata, mely két irányban mozoghat az egy irányban végtelen szalagon és tud írni is. Működéskor elolvas egy szimbólumot, majd keres hozzá egy szabályt és eszerint felülírja, utána továbblép.  
**Church-tézis:** Minden formalizálható probléma, ami megoldható algoritmussal, az megoldható Turing-géppel is.  
**Megállási probléma:** A Turing-gép megállási problémájának eldöntésére nem adható eljárás  
**Algoritmikusan eldönthető probléma:** Olyan problémák, amelyeket reprezetáló nyelvek rekurzívan felsorolhatók, azaz van olyan algoritmus, amely a nyelv szavait felsorolja.

**Logikai függvények megadása:** szöveges, táblázatos, halmazos, logikai vázlat, algebrai  
**KNF:** Elemi diszjunkciók konjunkciója  
**DNF:** Elemi konjunkciók diszjunkciója  
**Logikai hálózatok: ??????????? Nem egyértelmű, hogy ide mi kell ???????????????**

1. **Tétel**

**Adatszerkezetek és algoritmusok:** sor, verem, hiányos mátrix, láncolt lista, bináris fák műveleteinek algoritmusai.  
**Operációs rendszerek:** Folyamatkezelés és -ütemezés. Memóriakezelés. Állománykezelés.  
**Formális nyelvek és automaták:** Az üresszó-lemma. Véges automata fogalma, fajtái, véges automaták determinisztikussá tétele.  
  
**Sor:** FIFO (push, pop) **Verem:** LIFO (push, pop, top, empty, size)  
**Hiányos mátrix:** n(n+1) /2+1 méretű, az értékes elemek után van még egy 0  
**Láncolt lista:** A tárolt elemek tárolják a következő elem címét  
**Bináris fa:** Minden elemnek legfeljebb két rákövetkező gyermeke van. Láncolt ábrázolás, tömbös ábrázolás (csak teljes és majdnem teljes fa) Bejárások: preorder, inorder, postorder

**Operációs rendszer:** Erőforráskezelés  
**Folyamatkezelés, ütemezés:** program -> memória = folyamat Ütemezés: prioritásos, időosztásos  
**Memóriakezelés:** Feladatai az adatterületek biztosítása, a memória megosztása a folyamatok között. Virtuális memória: windows lapozófájl, linux swap partíció  
**Állománykezelés:** partíciók, fájlrendszerek, jogosultságkezelés, fájltípusok

**Üresszó lemma:** Minden környezetfüggetlen grammatikához megadható olyan G’ környezetfüggetlen nyelvtan, hogy a két grammatika ugyanazt a nyelvet generálja és G’ környezetfüggő is.  
**Automata:** Különböző időpillanatokban történő eseményekre reagál, minközben a belső állapotát a szabályoknak megfelelően változtatja. Egy automata véges, ha az állapothalmaz, a bemenő jelhalmaz és a kimenő jelhalmaz is véges. Bármely véges nemdeterminisztikus automatához létezik vele ekvivalens determinisztikus véges automata.

1. **tétel**

***Objektum-orientált programozás:*** *OOP, típusok és konverzióik, operátorok, utasítások. Metódusok, osztálykészítés, láthatóság, konstruktor.****Formális nyelvek és automaták:*** *Ábécé, szó, nyelv, nyelvtan fogalma. Chomsky-féle nyelvtani osztályok és az általuk generált nyelvosztályok tartalmazási hierarchiája.****Számítógép-architektúrák:*** *logikai áramkörök, kombinációs logikai hálózatok (fél és teljes összeadó, multiplexer, demultiplexer, dekóder).*

**OOP:** Objektumokból és a közöttük történő kölcsönhatásokból épül fel a program. Egységbezárás, absztrakció (csak a szükséges infó látszik a külvilág felé), öröklés, polimorfizmus.  
**??? Típusok és konverzióik:** primitív és referencia típusok. Implicit és explicit konverzió  
**??? Operátorok:** egy, kettő vagy három operandusú; aritmetikai, relációs, léptető, bitműveletek  
**??? Utasítások:** Egy futtatási egység, pontosvesszővel lezárva  
**Metódusok:** az osztályok tagfüggvényei  
**Osztálykészítés:** class kulcsszó, blabla  
**Láthatóság:** Az osztályon kívülre mutatni kívánt dolgok meghatározása. Pl.: private, public, protected, internal, stb.  
**Konstrukror:** Inicializálja a példányt. New operator. Több konstruktor is lehet.

**Ábécé:** Szimbólumok tetszőleges, nem üres véges halmaza.  
**Szó:** Az ábécé betűiből felírható véges hosszúságú sorozatok  
**Üresszó:** Olyan szó, amelynek egyetlen betűje sincs  
**Nyelv:** Szavak halmaza  
**Generatív nyelvtan:** A G=(N,T,S,H) rendezett négyest generatív nyelvtannak nevezzük.  
**Chomsky-féle nyelvtani osztályok 🡪 tartalmazási hierarchia:**   
- 0-ás típusú/általános, mondatszerkezetű: Minden szabály nemterminálisok és terminálisokból álló párokból áll, melynek bal oldala tartalmaz legalább egy nemterminálist. 🡪Turing-gép  
- 1-es típusú/környezetfüggő: Minden szabály vagy p1Ap2 -> p1qp2 alakú. Vagy S->lambda, de akkor az S nem szerepelhet egyik szabály jobb oldalán sem. 🡪 Lineárisan korlátolt Turing-gép  
- 2-es típusú/környezetfüggetlen: Minden szabály bal oldalán nemterminális, jobb oldalán nemterminális vagy terminális található 🡪 Veremtáras automata  
- 3-as típusú/reguláris: Minden szabály bal oldalán nemterminális, jobb oldalán egy terminális vagy egy terminális és nemterminális pár található 🡪Véges automata

**Logikai áramkör:** Olyan áramkör, ami logikai alapműveletekből épül fel.  
**Logikai kapuk:** AND, OR, NOT, NAND, NOR, XOR, XNOR  
**Kombinációs áramkörök:** Többszörös bemeneteik és többszörös kimeneteik vannak  
**multiplexer:** több bemenetet tesz egy kimenetre, időeltolódással egyesíteni tudja a bemeneteket  
**demultiplexer:** egy bemenetet tesz több kimenetre, időeltolódással szét tudja választani a bemenetet a vezérlőjel alapján  
**dekóder:** több bemenet kombinációja alapján aktivál egy kimenetet  
**félösszeadó:** egy bites egészek összeadását végzi, a átvitt bitet nem kezeli, ezt egy másik félösszeadóval kiegészítve egy **teljes összeadót** kapunk.

1. **tétel**

***Objektum-orientált programozás:*** *Öröklődés, túlterhelés, polimorfizmus. Kivételkezelés****.  
Adatbázis-rendszerek:*** *Nézettáblák relációs adatbáziskezelőkben. Indexelés a táblákon – mikor használjuk? Az adatbázis-tervezés elmélete: funkcionális függőségek és normalizáció – Boyce-Codd normálforma (BCNF). Anomáliák nem normalizált adatbázissémák esetén. Az E/K modell és átfordítása relációs adatmodellé.****Programozási technológiák:*** *Nyomkövetés és hibakeresés, egységtesztelés, naplózás. Kollekciók használata, relációs adatbázisok kezelése OO programozási nyelvekben.*

**Kivételkezelés:** Exception osztályból származtathatóak. Try, catch, finally. Throw saját kivétel adása, ha önmagában adjuk meg akkor fentebbi kontextusba dobja a kivételt.

**Nézettábla:** Lekérdezések eredményének virtuális táblaként való használata. CREATE VIEW, DROP VIEW  
**Index:** A keresés gyorsítására használható  
**Funkcionális függés:** Ha egy tábla valamelyik mezőjében lévő érték meghatározza egy másik mező értékét.  
**Részleges funkcionális függés:** Ha egy mező értéke az összetett kulcs egy részétől függ csak  
**Tranzitív függés:** minden nem kulcs függ a kulcstól, de van olyan mező, ami mástól is függ  
**Normalizálás:** Redundanciamentes szerkezet kialakítása  
**1NF:** Egyik attribútum sem összetett adat.  
**2NF:** Nem tartalmaz részleges függőséget.  
**3NF:** Egyetlen másodlagos attribútum sem függ tranzitívan  
**BCNF:** Egyáltalán nem tartalmaz tranzitív függőséget.  
**Anomáliák:** Beszúrási – egy másik, logikailag nem kapcsolódó rész beszúrását is igényli; Törlési – egy másik, logikailag nem kapcsolódó adatcsoportot is elveszítünk; Módosítási – több helyen történő módosítást is igényel  
**EK modell -> relációs adatmodell:** Egyedek -> reláció, kapcsolatok -> kapcsolótábla  
  
**Nyomkövetés, hibakeresés:** utasítások követése a program futása közben (debug), változófigyelés, breakpoint.  
**Egységtesztelés:** A program legkisebb egységének tesztelése. Az elvárt kimenetek ellenőrzése. A tesztelési hierarchia legalsó tagja. TDD.  
**Naplózás:** A program futása közben megjeleníthetőek a fontosabb információk. Használható hibakeresésre.  
**Kollekciók: (**LinkedList, ArrayList, Set, Map) A kollekcióknak saját metódusai vannak, amellyel kezelhetjük azokat.  
  
**relációs adatbázisok kezelése OO programozási nyelvekben:** ORM-ek (Hibernate, LINQ, Eloquent, Mongoose, TypeORM)

1. **tétel**

***Objektum-orientált programozás:*** *Osztály- és példány inicializálás, konstruktor. Interfészek. Generikus programozás, összetett adatszerkezeteket implementáló osztályok és fontosabb műveleteik.****A rendszerfejlesztés technológiája:*** *rendszerfejlesztési modellek, tervezés, tesztelés, UML osztálydiagram. Verziókezelés.****Számítógép-architektúrák:*** *A mikroelektronika alapjai (félvezetők, dióda, tranzisztorok, fajtái és az általuk megvalósítható kapuk). A CPU és felépítése. Integrált áramkörök. Memóriák fajtái, csoportosításuk.*

**Osztály- és példány inicializálás:** Alapértelmezett vagy meghatározott értékek. Sorrendben inicializálja. Blabla  
**Konstruktor:** A példány inicializálása.  
**Interfész:** Absztrakt osztály. Egy interfészt implementáló osztálynak meg kell valósítani mindent, amit az interfész meghatároz.  
**Generikus programozás:** Általánosítás. Típusfüggetlen kód írása. Újrafelhasználhatóság. Pl.: listák, Promise<Type>, stb. Általános típus megadásakor felveszi az elsőnek kapott típust és utána az kikényszeríthető vele.  
**Összetett adatszerkezetek:** Nyelvenként változó. Pl.: kollekciók, listák, halmazok, asszociatív tömbök

**Rendszerfejlesztési modellek:** vízesés – egymásra épülő fázisok; evolúciós – megvalósítás -> módosítás; inkrementális – fontossági sorrend a fejlesztésben; agilis/scrum - sprint, daily meeting, product owner, scrum master, retrospective, code review  
**Tervezés:** követelmény -> architecktúra -> felület -> komponensek -> adatstruktúra -> algoritmusok  
**Tesztelés:** unit -> integrációs (unitok együttműködése) -> rendszer (teljesítmény, terhelés, megbízhatóság, hibakezelés tesztje) -> validációs (felhasználói elvárások, valós környezet)  
**UML:** Általános célú modellezési nyelv. Több különböző típusú diagram készíthető. Az osztálydiagram statikus modell. A használt osztályokat mutatja az attribútumokkal együtt. Szabályosan megszerkesztett UML-ből bizonyos nyelveknél kód generálható.  
**Verziókezelés:** centralizált (SVN) és decentralizált/elosztott (GIT).

**Félvezető:** gyengén vezeti az áramot de nem szigetel  
**dióda:** egyenirányítás, speciális dióda pl a LED  
**tranzisztor:** jelek kapcsolása, feszültség stabilizálása, jelerősítés, logikai áramkörökhöz **CPU:** ALU – alapvető számítások, része az FPU(lebegőpontos) ; CU – vezérlőegység; regiszter – 32/64bit, éppen futó műveletek adatait tárolja; buszkontroller – kapcsolat a regiszter és más tárolók közt; cache - gyorsítótár  
**IC:** áramkörök kisméretű félvezető lapkán, kicsi méret, kisebb fogyasztás, megbízhatóbb  
**memória:** ROM, RAM (DDR, frekvencia, időzítés), virtuális memória.