Digitális Kultúra Érettségi – Mintatételek (B)

- Az érettségi vizsgán 1-től kezdődik a listák (tömbök!) indexelése.
- Ha kapsz algoritmust, minden esetben lekódolhatod, még ha a feladat nem is kéri. (Kódold is, az segít.)
- Nem minden feladat algoritmizálásra épül. A feladatok kb. az alábbiak szerint oszlanak el egy tételsorban (kb. 25-25-25-25% arányban):
    - Egyszerű gyakorlati feladat (Pl.: készíts fej vagy írást szimuláló programot, dobókocka dobást szimuláló programot, stb…). Ezek pont olyanok, mint amik a gyakorlati vizsgára kellettek, csak egyszerűbbek. Ilyeneket nem írtam a tételsorba, ezeket gyakoroltátok eleget a gyakorlati vizsgára készülés során.
    - Algoritmust javító feladatok: Előfordul, hogy egy nem túl bonyolult algoritmust kell javítani, és semmi más dolgod nincs. Ilyenkor több hiba is van az algoritmusban, akár 6-7 is. Általában az alábbi dolgok lehetnek hibásak:
        - Változók értékadása
        - Változók összekeverése
        - Listák indexelése (érettségin pszeudokódban 1-től, de programozáskor 0-tól)
        - Ciklus vagy elágazás feltételek (relációsjel, hossz helyett hossz – 1, stb…)
    - Algoritmust javító és algoritmust leprogramozó feladat. Ha a feladat nem csak a javítást, hanem az algoritmus leprogramozását is kéri, akkor ebben általában kevesebb hiba van (2-3).
    - Algoritmust leprogramozó feladat. Kapsz egy nehéz algoritmust, nem hibás, csak le kell kódold, és el kell tudd magyarázni, hogy mit csinál.
- A tételsorom nem fókuszál a leprogramozásra, hiszen arra a gyakorlati rész miatt már készültél. Persze magadnak ezeket bármikor lekódolhatod, de nagyon fontos, hogy úgy menj el vizsgára, hogy pszeudokódot maximálisan tudsz olvasni, értelmezni, átírni.
- Fontos, hogy ezek MINTA tételek. NEM pont ezeket fogod kapni, hasonlóak előfordulhatnak. De inkább arra koncentrálj, hogy az algoritmikus gondolkodást, algoritmusok elemzését szokd meg. Erről szól főként a szóbeli vizsga.

1. tétel: Hibakeresés egy összegzési algoritmusban

Pszeudokód:

```
összeg := 0
LISTA := [1, 2, 3, 4, 5, 6]
Ciklus i := 1-től LISTA hossz+1-ig:
    összeg := összeg + LISTA[i + 1]
KIÍR összeg
```

Feladat: A pszeudokód a lista elemeinek összegét kalkulálja. Találd meg és javítsd ki a hibákat!

Megoldás:

```
összeg := 0
LISTA := [1, 2, 3, 4, 5, 6]
Ciklus i := 1-től LISTA hosszáig:
    összeg := összeg + LISTA[i]
KIÍR összeg
```

2. tétel: Pszeudokód kielemzése - Maximumkeresés

Pszeudokód:

```
MAX := LISTA[1]
Ciklus i := 2-től LISTA hossz-ig:
    HA LISTA[i] > MAX:
        MAX := LISTA[i]
KIÍR "A legnagyobb elem: ", MAX
```

Feladat: Magyarázd el, hogyan működik ez az algoritmus.

Megoldás:
Inicializáljuk a MAX változót, beállítva a listánk (tömbünk) 1. elemére. Ehhez szükséges, hogy a listánknak legalább 1 eleme legyen, ekkor ez az elem a maximum is. Ez után végigjárjuk a listát a második elemtől kezdve az utolsó elemig. A ciklusmagon belül összehasonlítjuk a lista i-edik elemét a MAX változóban tárolt értékkel. Amennyiben ez az érték meghaladja a MAX értékét, úgy a MAX változó értékének beállítjuk a lista i-edik („aktuális”) elemét. A ciklus végeztével kiírjuk a lista legnagyobb elemét.

3. tétel: Pszeudokód írás - Két lista metszetének meghatározása

Feladat: Írj pszeudokódot, amely meghatározza két számokból álló lista metszetét.

Megoldás:

```
LISTA1 := [1, 2, 3, 4, 5]
LISTA2 := [3, 4, 5, 6, 7]
METSZET := []
DB := 0
Ciklus i := 1-től LISTA1 hossz-ig:
    Ciklus j = 1-től Lista2 hossz-ig:
      HA LISTA1[i] = LISTA2[j]:
        METSZET[DB] := LISTA1[i]
        DB := DB + 1
KIÍR "Metszet: ", METSZET
```

4. tétel: Típusalgoritmus leírása - Másolás

Feladat: Magyarázd el a Másolás típusalgoritmusát és írj hozzá pszeudokódot.

Megoldás:

```
EREDETI := [3, 5, 2, 6, 5, 2]
ÚJ := []
Ciklus i := 1-től EREDETI hossz-ig:
  HA EREDETI[i] megfelel egy Tulajdonságnak:
    ÚJ hozzáad EREDETI[i]
KIÍR ÚJ
```

5. tétel: Hibakeresés egy eldöntési algoritmusban

Pszeudokód:

```
  Talált := IGAZ
  Ciklus i := 1-tól LISTA hossza-ig:
     HA LISTA[i] = KeresettElem:
        Talált := IGAZ
```

```
        Kilépés a ciklusból
    HA Talált = HAMIS:
        KIÍR "Az elem nem található."
```

Feladat: Az algoritmus eldönti, hogy a `KeresettElem` megtalálható-e a listában. Hibás-e az algoritmus?

Megoldás:

A Talált változót eleinte IGAZ-ra állítjuk, majd a ciklusban, ha megtaláljuk az elemet, újra IGAZ-ra állítjuk, és kilépünk. Ha a ciklus végén Talált = HAMIS, írjuk ki, hogy nem található.

Azonban, a Talált változót soha nem állítjuk HAMIS-ra. Tehát, ha a lista nem tartalmazza a keresett elemet, a találatunk soha nem lesz HAMIS. Ez hibás, hiszen az algoritmus minden bemenet esetén azt fogja jelezni, hogy az elem megtalálható a listában.

Helyesen:

```
Talált := HAMIS
Ciklus i = 1-től LISTA hossz-ig:
  Ha LISTA[i] = KeresettElem:
    Talált := IGAZ
    Kilépés a ciklusból
HA talált = HAMIS:
  KIÍR „Az elem nem található.”
KÜLÖNBEN:
  KIÍR „Az elem megtalálható a listában.”
```

6. tétel: Pszeudokód kielemzése - Páros és páratlan számok szétválasztása

Pszeudokód:

```
    LISTA = [4, 2, 7, 6, 5, 8]
    PÁROSOK := []
    PÁRATLANOK := []
    Ciklus i := 1-től LISTA hossz-ig:
        HA LISTA elem MOD 2 = 0:
            PÁROSOK hozzáad LISTA elem
        KÜLÖNBEN:
```

```
        PÁRATLANOK hozzáad LISTA elem
    KIÍR "Páros számok: ", PÁROSOK
    KIÍR "Páratlan számok: ", PÁRATLANOK
```

Feladat: Magyarázd el, hogyan választja szét az algoritmus a páros és páratlan számokat.

Megoldás:

Az algoritmusunk először két üres listát hoz létre, PÁROSOK és PÁRATLANOK néven. Ezekbe fognak kerülni a páros, illetve páratlan számok. Ez után végigmegyünk A LISTA elemein, hogy minden elemet sorban meg tudjunk vizsgálni. Mivel a paritásvizsgálat két kimenetet eredményezhet, egy elágazással megnézzük, hogy az i-edik elem 2-vel osztva 0 maradékot ad-e, azaz páros-e. Ha igen, a PÁROSOK közé tesszük. Ellenkező esetben a PÁRATLAN-ok közé. A ciklus végeztével kiírjuk mindkét lista elemeit.

7. tétel: Feladatleírás alapján pszeudokód - LKT meghatározása

Feladat: Írj pszeudokódot, amely kiszámítja két szám legkisebb közös többszörösét.

```
BE: A, B
a := A
b := B

Ciklus AMÍG a ≠ b:
    HA a < b:
        a := a + A
    KÜLÖNBEN:
        b := b + B

KI: "A legkisebb közös többszörös:", a
```

8. tétel: Típusalgoritmus leírása és pszeudokód

Feladat: Magyarázd el az Unio algoritmusát és írj hozzá egy pszeudokódot.

Az unio célja két halmaz/tömb/lista egyesítése, olyan új tömb létrehozása, ami mindkettő elemeit tartalmazza, de ismétlődés nélkül.

```
LISTA1 := [3, 5, 3, 6]
LISTA2 := [3, 6, 7, 5]
UNIO := []
Ciklus i := 1-től LISTA1 hossza-ig:
    UNIO hozzáad LISTA1[i]
Ciklus i := 1-től LISTA2 hossza-ig:
    HA LISTA[i] nincs benne UNIO-ban:
        UNIO hozzáad LISTA2[i]
```

9. tétel: Hibakeresés egy rendezési algoritmusban (Buborékrendezés)

Pszeudokód:

```
    Ciklus i := 1-tól LISTA hossza-ig:
        Ciklus j := 1-tól LISTA hossza - i -ig:
            HA LISTA[j] > LISTA[j+1]:
                CSERE LISTA[j] és LISTA[j+1]
```

Feladat: A pszeudokód célja egy lista buborékrendezéssel való rendezése. Találd meg a hibát!

Megoldás:

```
Ciklus i := 1-tól LISTA hossza - 1-ig:
    Ciklus j := 1-tól LISTA hossza - i-ig:
        HA LISTA[j] > LISTA[j+1]:
            TEMP := LISTA[j]
            LISTA[j] := LISTA[j + 1]
            LISTA[j + 1] := TEMP
```

10. tétel: Pszeudokód kielemzése - Fibonacci-sorozat

Pszeudokód:

```
FIBONACCI := [0, 1]
Ciklus i := 3-tól N-ig:
    ÚjElem := FIBONACCI[i-1] + FIBONACCI[i-2]
    FIBONACCI hozzáad ÚjElem
KIÍR FIBONACCI
```

Feladat: Programozd le, és magyarázd el, hogyan generálja ez az algoritmus a Fibonacci-sorozat első N elemét.

Megoldás:

Adott a Fibonacci-sorozat első két eleme, 0 és 1 egy listában. Indítunk egy ciklust ami 3-tól N-ig megy, hiszen az első 2 elem már megvan. Minden iterációban az új elemet úgy képezzük, hogy a 2 előtte lévő elemet összeadjuk, és ezt az új elemet hozzáadjuk a FIBONACCI listához. A ciklus végén kiírjuk a lista elemeit. {Példa szemléltetéssel…}

Python program:

```
N = 7
FIBONACCI = [0, 1]
for i in range(2, N):
    ÚjElem = FIBONACCI[i-1] + FIBONACCI[i-2]
    FIBONACCI.append(ÚjElem)
print(FIBONACCI)
```

11. tétel: Feladatleírás alapján pszeudokód - Tömb elemeinek átlaga

Feladat: Írj pszeudokódot, amely kiszámítja egy számokból álló tömb elemeinek átlagát.

Megoldás:

```
BE: TÖMB[1..N]
OSSZEG := 0
Ciklus i := 1-től N-ig:
    OSSZEG := OSSZEG + TÖMB[i]
ATLAG := OSSZEG / N {opcionális kerekítéssel}
KIÍR ATLAG
```

12. tétel: Típusalgoritmus leírása és pszeudokód - Összegzés

Feladat: Magyarázd el az összegzés algoritmusának működését és készíts hozzá egy pszeudokódot, és szemléltesd egy példával!

Megoldás:
BE: TÖMB[1..N]

OSSZEG := 0

Ciklus i := 1-től N-ig:

    OSSZEG := OSSZEG + TÖMB[i]

KIÍR OSSZEG

Példa:

TÖMB = [3, 5, 7]

N = 3

OSSZEG = 0

i=1 -> OSSZEG = 0 + 3 = 3

i=2 -> OSSZEG = 3 + 5 = 8

i=3 -> OSSZEG = 8 + 7 = 15

Kimenet: 15

13. tétel: Hibakeresés egy kiválasztási algoritmusban

Pszeudokód:

```
LEGNAGYOBB := LISTA[1]
INDEX := 2
Ciklus i := 1-től LISTA hossz+1-ig:
    HA LISTA[i] > LEGNAGYOBB:
        LEGNAGYOBB := i
KIÍR INDEX
```

Feladat: Az algoritmus a legnagyobb elem indexét próbálja megkeresni a listában. Javítsd ki a hibát!

Megoldás:

```
LEGNAGYOBB := LISTA[1]
INDEX := 1
Ciklus i := 2-től LISTA hosszáig:
   HA LISTA[i] > LEGNAGYOBB:
      LEGNAGYOBB := LISTA[i]
      INDEX := i
KIÍR  INDEX
```

14.tétel: Pszeudokód kielemzése - Prímszámok keresése

Pszeudokód:

```
   PRÍMEK := []
   Ciklus i := 2-től N-ig:
      Prím := IGAZ
      Ciklus j := 2-től i DIV 2-ig:
         HA i MOD j = 0:
            Prím := HAMIS
            Ciklus törése
      HA Prím:
         PRÍMEK hozzáad i
   KIÍR PRÍMEK
```

Feladat: Programozd le és magyarázd el, hogyan azonosítja az algoritmus a prímszámokat 2 és N között.

Készítünk egy listát, amibe a prímszámokat tervezzük kimenteni. Ez után indítunk egy ciklust amivel 2-től N-ig elszámolunk (hiszen a 2 a legelső prímszám), és minden számot először PRÍM-nek állítunk. Az algoritmus fő működési elve az, hogy azt vizsgáljuk, hogy a prím tulajdonság mikor romlik el. Ezt osztókkal ellenőrizzük, minden számnak a szám feléig ellenőrizzük az osztóit, hiszen ha két szám szorzatára lehet bontani, az egyik fele a szorzatnak kisebb lesz úgyis, mint a kiindulási számunk fele. Ha találunk egy `j` osztót `i` számhoz, 2 és a `i` fele között, akkor `i` prím tulajdonsága elromlik, ő nem lesz prímszám. A ciklus törése optimalizálási cél miatt van, hogy ne nézzen több osztót `i`-hez, hiszen már úgyis elromlott a prím tulajdonság.

```
N = 10
primek = []
for i in range(2, N+1):
    prim = True
    for j in range(2, i // 2 + 1):
        if i % j == 0:
            prim = False
            break
    if prim:
        primek.append(i)
print(primek)
```

15. tétel: Feladatleírás alapján pszeudokód - Tömb fordított sorrendbe rendezése

Feladat: Írj pszeudokódot egy tömb elemeinek fordított sorrendbe történő rendezéséhez.

```
BE: TÖMB[1..N]
BAL := 1
JOBB := N
AMÍG BAL < JOBB:
    TEMP := TÖMB[JOBB]
    TÖMB[JOBB] := TÖMB[BAL]
    TÖMB[BAL] := TEMP
    BAL := BAL + 1
    JOBB := JOBB - 1
KIÍR TÖMB
```

16. tétel: Típusalgoritmus leírása és pszeudokód - Kiválogatás

Feladat: Magyarázd el a kiválogatás algoritmusát, és írj hozzá egy pszeudokódot, amely kiválogatja a páratlan számokat egy tömbből.

Megoldás:

A kiválogatás algoritmusa arra szolgál, hogy egy sorozatból (lista, tömb) bizonyos feltételnek megfelelő elemeket külön gyűjtsünk. Létrehozunk egy KIVÁLOGATOTT tömböt, és az összes elem bejárása során minden egyes elemre megvizsgálunk egy feltételt, pl hogy páratlan-e a szám. Amennyiben ez a feltétel teljesül hozzáadjuk a listához az aktuális elemet.

```
BE: TÖMB[1..N]

PÁRATLANOK := üres lista

Ciklus i := 1-től N-ig:

   HA TÖMB[i] MOD 2 ≠ 0:

      PÁRATLANOK hozzáad TÖMB[i]

KIÍR PÁRATLANOK
```

17. tétel: Hibakeresés egy megszámlálás algoritmusban

Pszeudokód:

```
   NULLÁK := 1

   LISTA := [0, 1, 0, 1, 1, 0, 1]

   Ciklus i := 1-tól LISTA hosszáig:

      HA LISTA[i] = 0:

         NULLÁK := NULLÁK + 1

   KIÍR "Nullák száma: ", NULLÁK - 1
```

Feladat: A kód a nullák számát próbálja megszámolni egy listában. Javítsd ki a hibát!

Megoldás:

```
NULLÁK := 0

LISTA := [0, 1, 0, 1, 1, 0, 1]

Ciklus i := 1-tól LISTA hosszáig:

   HA LISTA[i] = 0:

      NULLÁK := NULLÁK + 1

KIÍR NULLÁK
```

18. tétel: Pszeudokód kielemzése - Többször előforduló elemek szűrése

Pszeudokód:

```
ELEMEK := [1, 2, 2, 3, 4, 4, 4, 5]

EGYEDI := []

Ciklus i := 1-től elemek hossza-ig:

    HA az ELEM nem szerepel az EGYEDI-ben:

        EGYEDI hozzáad ELEM

KIÍR EGYEDI
```

Feladat: Programozd le és magyarázd el, hogyan szűri ki az algoritmus a többször előforduló elemeket.

Megoldás:

Létrehozunk egy kezdő listát, ami kezdetben üres. Ebbe pedig egy ciklus segítségével hozzáadjuk azokat az elemeket amik nem szerepelnek még az EGYEDI listában.

ELEMEK = [1, 2, 2, 3]

EGYEDI = []

1 -> nincs benne -> EGYEDI = [1]

2 -> nincs benne -> EGYEDI = [1, 2]

2 -> már benne -> kihagyjuk

3 -> nincs benne -> EGYEDI = [1, 2, 3]

```
elemek = [1, 2, 2, 3, 4, 4, 4, 5]

egyedi = []

for elem in elemek:

    if elem not in egyedi:

        egyedi.append(elem)

print("Egyedi elemek:", egyedi)
```

19. tétel: Feladatleírás alapján pszeudokód: Legnagyobb és legkisebb elem keresése

Feladat: Írj pszeudokódot, amely megtalálja és kiírja egy tömb legnagyobb és legkisebb elemét.

```
BE: TÖMB[1..N]

LEGKISEBB := TÖMB[1]

LEGNAGYOBB := TÖMB[1]

Ciklus i := 2-től N-ig:

   HA TÖMB[i] < LEGKISEBB:

      LEGKISEBB := TÖMB[i]

   HA TÖMB[i] > LEGNAGYOBB:

      LEGNAGYOBB := TÖMB[i]

KIÍR LEGKISEBB

KIÍR LEGNAGYOBB
```

20. tétel: Típusalgoritmus leírása és pszeudokód: Eldöntés

Feladat: Magyarázd el az eldöntés algoritmusát, és írj hozzá egy pszeudokódot, amely eldönti, hogy egy adott szám szerepel-e egy listában.

Megoldás:

Az eldöntés algoritmus arra szolgál, hogy megvizsgáljuk, teljesül-e egy adott feltétel egy sorozat valamelyik elemére, és IGAZ vagy HAMIS értéket adjon eredményül. Tipikus példája hogy egy bizonyos szám szerepel-e egy adott listában. Alapból feltételezzük, hogy a szám nincs a listában, és amint ez a tulajdonság elromlik, mert megtaláljuk, úgy a logikai változó értékét IGAZ-ra állítjuk.

```
BE: LISTA[1..N], KeresettSzam

Talált := HAMIS

Ciklus i := 1-től N-ig:

   HA LISTA[i] = KeresettSzam:

      Talált := IGAZ

      Szakítsd meg a ciklust

HA Talált:

   KIÍR KeresettSzam, " megtalálható a listában."

KÜLÖNBEN:

   KIÍR KeresettSzam, " nem található a listában."
```