BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA-INFORMATIKA KAR

MATEK-IINFO verseny BBTE – 2021

Informatika írásbeli

FONTOS MEGJEGYZÉS:
Más pontosítások hiányában feltételezhetjük, hogy az aritmetikai műveleteket végtelen adattípusokon végezzük, vagyis nincs túlcsordulás és alulcsordulás.
Minden sorozatot 1-től számozunk

1. Legyen a következő kifejezés, ahol ***a*** természetes szám.

```
((a < 4) VAGY (a < 5)) ÉS (a > 2)
```

Az ***a*** mely értékeire lesz a kifejezés értéke $igaz$?

A. ***a*** = 3
B. ***a*** = 4
C. ***a*** = 2
D. A kifejezés értéke soha nem lesz $igaz$.

2. Az alábbi algoritmusnak bemeneti paraméterei az ***n*** egész szám (1 ≤ ***n*** ≤ 10000) és az ***n*** elemű ***v*** sorozat, amelynek elemei nullától különböző természetes számok (***v***[1], ***v***[2], …, ***v***[***n***]).

```
Algoritmus f(v, n):
    x ← 0
    Minden i ← 1, n végezd el
        c ← v[i]
        Amíg c MOD 3 = 0 végezd el
            x ← x + 1
            c ← c DIV 3
        vége(amíg)
    vége(minden)
    visszatérít x
Vége(algoritmus)
```

A következő állítások közül melyek igazak?

A. Az algoritmus visszatéríti a ***v*** sorozat 3-mal osztható elemeinek darabszámát
B. Az algoritmus visszatéríti azt a legnagyobb ***k*** számot, amelyre ***v***[1] * ***v***[2] * … * ***v***[***n***] osztható 3^k-nal
C. Az algoritmus visszatéríti azt a legnagyobb ***k*** számot, amelyre ***v***[1] + ***v***[2] + … + ***v***[***n***] osztható 3^k-nal
D. Az algoritmus visszatéríti a ***v*** sorozat 3-mal osztható elemeinek összegét

3. Legyen a következő kifejezés, ahol ***x*** pozitív természetes szám.

```
(x MOD 2) + ((x + 1) MOD 2)
```

A következő állítások közül melyek igazak?

A. A kifejezés értéke egyenlő 1-gyel bármely ***x*** pozitív természetes szám esetén.
B. A kifejezés értéke akkor és csakis akkor egyenlő 1-gyel, ha ***x*** páros szám.
C. A kifejezés értéke akkor és csakis akkor egyenlő 1-gyel, ha ***x*** páratlan szám.
D. Létezik olyan ***x*** természetes szám, amelyre a kifejezés értéke szigorúan nagyobb, mint 1.

4. Legyen az alábbi `feldolgozás(x, n)` algoritmus, amelynek bemeneti paraméterei az ***n*** egész szám ($1 \leq n \leq 10000$) és az ***n*** elemű ***x*** sorozat, amelynek elemei nullától különböző valós számok (***x***[1], ***x***[2], …, ***x***[***n***]).
A `/` műveleti jel valós osztást jelent (például `3 / 2 = 1,5`).

```
Algoritmus feldolgozás(x, n):
    p ← 1
    Minden k ← 1, n - 1 végezd el
        p ← p + 1
        Minden i ← 1, n - 1 végezd el
            Ha x[i] > x[i + 1] akkor
                x[i] ← x[i] * x[i + 1]
                x[i + 1] ← x[i] / x[i + 1]
                x[i] ← x[i] / x[i + 1]
            vége(ha)
        vége(minden)
    vége(minden)
    n ← p
Vége(algoritmus)
```

A következő állítások közül melyek írják le az ***x*** sorozat módosulását a `feldolgozás(x, n)` algoritmus meghívásának eredményeképpen?

A. Az ***x*** sorozat elemei változatlanok maradnak
B. Az ***x*** sorozat elemei csökkenő sorrendbe fognak kerülni
C. Az ***x*** sorozat elemei növekvő sorrendbe fognak kerülni
D. Az ***n*** változó értéke csökkenni fog eggyel

5. Legyen az alábbi `számol(a, n)` algoritmus, amelynek bemeneti paraméterei az ***n*** egész szám ($1 \leq n \leq 10000$) és az ***n*** elemű ***a*** sorozat, amelynek elemei természetes számok (***a***[1], ***a***[2], …, ***a***[***n***]).

```
Algoritmus számol(a, n):
    Ha n = 0 akkor
        visszatérít 0
    különben
        visszatérít a[n] * (a[n] MOD 2) + számol(a, n - 1)
    vége(ha)
Vége(algoritmus)
```

Az ***n*** és ***a*** mely értékeire lesz a `számol(a, n)` algoritmus visszatérítési értéke 10?

A. ***n*** = 4, ***a*** = (2, 4, 7, 5)
B. ***n*** = 6, ***a*** = (3, 1, 2, 5, 8, 1)
C. ***n*** = 6, ***a*** = (2, 4, 5, 3, 8, 5)
D. ***n*** = 7, ***a*** = (1, 1, 2, 1, 1, 1, 3)

6. Legyen az alábbi `számol(v, n)` algoritmus, amelynek bemeneti paraméterei az ***n*** egész szám ($1 \leq \boldsymbol{n} \leq 10000$) és az ***n*** elemű ***v*** sorozat, amelynek elemei természetes számok (***v***[1], ***v***[2], …, ***v***[***n***]).

```
Algoritmus számol(v, n):
    m ← 0
    x ← 0
    s ← 0
    Minden i ← 1, n végezd el
        s ← s + v[i]
        m ← m + (s MOD 2 + x) MOD 2
        x ← s MOD 2
    vége(minden)
    visszatérít m
Vége(algoritmus)
```

A következő állítások közül melyek igazak?

A. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozatban található páratlan számok összegét
B. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozatban található páros számok összegét
C. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozatban található páratlan számok darabszámát
D. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozatban található páros számok darabszámát

7. Legyen a `bűvös(x)` algoritmus, ahol ***x*** természetes szám ($1 \leq \boldsymbol{x} \leq 32000$).

```
Algoritmus bűvös(x):
    bal ← 1
    jobb ← x
    Amíg bal ≤ jobb végezd el
        közép ← (bal + jobb) DIV 2
        Ha közép * közép = x akkor
            visszatérít igaz
        vége(ha)
        Ha közép * közép < x akkor
            bal ← közép + 1
        különben
            jobb ← közép – 1
        vége(ha)
    vége(amíg)
    visszatérít hamis
Vége(algoritmus)
```

A következő állítások közül melyek igazak?

A. Az algoritmus ellenőrzi, hogy létezik-e ***x***-nél kisebb négyzetszám.
B. Az algoritmus megszámolja az ***x*** szám prímszám osztóit.
C. Az algoritmus ellenőrzi, hogy az ***x*** szám prímszám-e.
D. Az algoritmus ellenőrzi, hogy az ***x*** szám négyzetszám-e.

8. Legyen a `mitCsinál(n)` algoritmus, ahol ***n*** természetes szám ($1 \leq \boldsymbol{n} \leq 10000$).

```
Algoritmus mitCsinál(n):
    a ← n
    b ← 0
    Amíg a ≠ 0 végezd el
      b ← b * 10 + a MOD 10
      a ← a DIV 10
    vége(amíg)
    Ha n = b akkor
      visszatérít igaz
    különben
      visszatérít hamis
    vége(ha)
Vége(algoritmus)
```

A következő állítások közül melyek igazak?

B. Az algoritmus vizsgálja, hogy az ***n*** szám prímszám-e.
C. Az algoritmus vizsgálja, hogy az ***n*** szám palindromszám-e.
D. Az algoritmus mindig *igaz* értéket térít vissza.
E. Az algoritmus vizsgálja, hogy az ***n*** szám osztható-e 10-zel.

9. Legyen a `számol(a, b)` algoritmus, ahol ***a*** és ***b*** természetes számok ($1 \leq \boldsymbol{a, b} \leq 10000$).

```
Algoritmus számol(a, b):
    x ← 1
    Minden i ← 1, b végezd el
        x ← (x MOD 10) * a
    vége(minden)
    visszatérít x
Vége(algoritmus)
```

A következő állítások közül melyek igazak?

A. Ha ***a*** = 2021 és ***b*** = 2021, az algoritmus által visszatérített érték 2021.
B. Ha ***a*** = 2021 és $1 \leq \boldsymbol{b} \leq 10000$, az algoritmus által visszatérített érték minden esetben 2021.
C. Ha ***a*** = 7777 és ***b*** = 2021, az algoritmus által visszatérített érték 7777.
D. Ha ***b*** = 2021 és $1 \leq \boldsymbol{a} \leq 10000$, az algoritmus által visszatérített érték minden esetben egyenlő ***a*** értékével.

10. Hány elem található egy ***n*** soros és ***n*** oszlopos négyzetes mátrix két átlóján ($10 \leq \boldsymbol{n} \leq 1000$)? Csak a különböző pozíciókon található elemeket számoljuk meg.

A. `2 * n`
B. `n * n`
C. `2 * n - 1`
D. `2 * n – (n MOD 2)`

11. Az alábbi logikai kifejezések közül melyeknek az értéke *igaz*, ha ***a*** = 1 és ***b*** = 0?

A. `NEM (((a > 0) ÉS (b < 1)) VAGY (a > 1))`
B. `((b > 0) ÉS (b < 1)) VAGY ((a > 0) ÉS (a < 2))`
C. `(NEM (a > b)) VAGY (NEM (b > 0))`
D. `(a > 0) VAGY ((b > 0) ÉS (b < 0)) VAGY (a < 1)`

12. A $\texttt{számol}_i\texttt{(e, n)}$, $1 \leq \boldsymbol{i} \leq 4$ algoritmusok bemeneti paraméterei az ***n*** soros és ***n*** oszlopos ***e*** mátrix (***e***[1][1], …, ***e***[1][***n***], ***e***[2][1], …, ***e***[***n***][***n***]) és az ***n*** természetes szám ($1 \leq \boldsymbol{n} \leq 1000$). Válasszátok ki az alábbi algoritmusok közül azokat, amelyek azoknak a $\texttt{számol}_i\texttt{(e, n)}$ algoritmusoknak a definícióját tartalmazzák, amelyeknek az eredménye különbözik a másik három algoritmus eredményétől, vagyis $\texttt{számol}_i\texttt{(e, n)} \neq \texttt{számol}_j\texttt{(e, n)}$ $\forall$ ***e***, ***n***, ***j***, $1 \leq \boldsymbol{j} \leq 4$, $\boldsymbol{i} \neq \boldsymbol{j}$ (***e*** és ***n*** megfelelnek a fenti specifikációnak).

A.

```
Algoritmus számol1(e, n):
    s ← 0
    Minden i ← 1, n végezd el
        s ← s + e[i][i]
    vége(minden)
    visszatérít s
Vége(algoritmus)
```

B.
```
Algoritmus számol2(e, n):
    s ← 0
    Minden i ← 1, n végezd el
        Minden j ← 1, n, végezd el
            Ha i = j akkor
                s ← s + e[i][j]
            vége(ha)
        vége(minden)
    vége(minden)
    visszatérít s
Vége(algoritmus)
```

C.
```
Algoritmus számol3(e, n):
    s ← 0
    i ← 1
    Amíg i ≤ n végezd el
        s ← s + e[i][i]
        i ← i + 1
    vége(amíg)
    visszatérít s
Vége(algoritmus)
```

D.
```
Algoritmus számol4(e, n):
    s ← 0
    Minden i ← 1, n végezd el
        Minden j ← i + 1, n, végezd el
            Ha i = j akkor
                s ← s + e[i][j]
            vége(ha)
        vége(minden)
    vége(minden)
    visszatérít s
Vége(algoritmus)
```

13. Legyen a `mitCsinál(a, b)` algoritmus, ahol $\boldsymbol{a}$ és $\boldsymbol{b}$ természetes számok ($1 \leq \boldsymbol{a} < \boldsymbol{b} \leq 10000$).

```
Algoritmus mitCsinál(a, b):
    m ← a
    Amíg b MOD m > 0 végezd el
        m ← m + 1
    vége(amíg)
    visszatérít m
Vége(algoritmus)
```

Mit fog visszatéríteni a `mitCsinál(47, 100)` hívás?

A. 48
B. 50
C. 3
D. 100

14. Legyen a `kiIr(n)` algoritmus, ahol $\boldsymbol{n}$ természetes szám ($0 \leq \boldsymbol{n} \leq 10000$).

```
Algoritmus kiIr(n):
    Ki: n
    Ha n > 0 akkor
        kiIr(n - 1)
        Ki: n
    vége(ha)
Vége(algoritmus)
```

Mit fog kiírni az algoritmus a `kiIr(4)` hívás eredményeképpen?

A. 432100123
B. 123401234
C. 1234004321
D. 432101234

15. Az alábbi számrendszerek közül melyekre érvényes a $232_{(x)} \leq 67_{(10)}$ reláció?

A. $\boldsymbol{x} = 5$
B. $\boldsymbol{x} = 3$
C. $\boldsymbol{x} = 4$
D. $\boldsymbol{x} = 6$

16. A `költöztetNullákat(a, n)` algoritmus bemeneti paraméterei az ***n*** elemű ***a*** sorozat, amelynek elemei egész számok (***a***[1], ***a***[2], …, ***a***[***n***]) és az ***n*** (1 ≤ ***n*** ≤ 10000) egész szám. Az algoritmus elköltözteti a 0 értékű elemeket a sorozat végére úgy, hogy megőrzi a 0-tól különböző elemek eredeti sorrendjét. Például, ha ***a*** = [4, 0, 2, 5, 1, 0, 7, 11, 0, 3], az algoritmus végrehajtása után ***a*** = [4, 2, 5, 1, 7, 11, 3, 0, 0, 0].

Az alábbi `költöztetNullákat(a, n)` algoritmusok közül melyek helyesek?

A.
```
Algoritmus költöztetNullákat(a, n):
    s ← igaz
    Amíg s = igaz végezd el
        s ← hamis
        Minden i ← 1, n - 1 végezd el
            Ha a[i] = 0 akkor
                tmp ← a[i]
                a[i] ← a[i + 1]
                a[i + 1] ← tmp
                s ← igaz
            vége(ha)
        vége(minden)
    vége(amíg)
Vége(algoritmus)
```

B.
```
Algoritmus költöztetNullákat(a, n):
    c ← 0
    Minden i ← 0, n végezd el
        Ha a[i] = 0 akkor
            c ← c + 1
        vége(ha)
    vége(minden)
    i ← n
    Amíg c > 0 végezd el
        a[i] ← 0
        i ← i - 1
        c ← c - 1
    vége(amíg)
Vége(algoritmus)
```

C.

```
Algoritmus költöztetNullákat(a, n):
    d ← 0
    i ← 1
    Amíg i + d ≤ n végezd el
        Amíg (i + d ≤ n) ÉS (a[i + d] = 0) végezd el
            d ← d + 1
        vége(amíg)
        Ha i + d ≤ n akkor
            a[i] ← a[i + d]
            i ← i + 1
        vége(ha)
    vége(amíg)
    Amíg i ≤ n végezd el
        a[i] ← 0
        i ← i + 1
    vége(amíg)
Vége(algoritmus)
```

D.

```
Algoritmus költöztetNullákat(a, n):
    i ← 1
    f ← n
    Amíg i < f végezd el
        Amíg (i < f) ÉS (a[i] ≠ 0) végezd el
            i ← i + 1
        vége(amíg)
        Amíg (i < f) ÉS (a[f] = 0) végezd el
            f ← f - 1
        vége(amíg)
        Ha i < f akkor
            tmp ← a[i]
            a[i] ← a[f]
            a[f] ← tmp
        vége(ha)
    vége(amíg)
Vége(algoritmus)
```

17. Legyen a következő sorozat: ***X*** = 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, …, amelyben minden ***n*** szám ***n*** darab egymásutáni pozíción szerepel. Tudva azt, hogy a sorozat első eleme az 1. pozíción található, mely pozíciókon jelennek meg a 21-es értékek?

A. A [210, 230] intervallumnak megfelelő pozíciókon
B. A [211, 231] intervallumnak megfelelő pozíciókon
C. A [212, 232] intervallumnak megfelelő pozíciókon
D. A [209, 229] intervallumnak megfelelő pozíciókon

18. Legyen az `f(a, b)` algoritmus, ahol ***a*** és ***b*** egész számok (-10000 ≤ ***a, b*** ≤ 10000).

```
Algoritmus f(a, b):
    Ki: "FMI"
    Ha (a = 0) vagy (b = 0) akkor
        visszatérít 1
    vége(ha)
    Ha a > b akkor
        visszatérít f(a - b * b, a * (a - b) - b * (a - b))
    vége(ha)
    Ha a ≤ b akkor
        visszatérít f(b - a * a, a * (a - b) - b * (a - b))
    vége(ha)
Vége(algoritmus)
```

Hányszor írja ki a program a `"FMI"` szöveget, ha a következő programrészletet hajtjuk végre:

```
f(f(3, 2), f(2, 3))
```

A. 8-szor
B. 6-szor
C. 3-szor
D. végtelenszer

19. Legyen a `mitCsinál(n, i)` rekurzív algoritmus, ahol ***n*** természetes szám ($2 \leq \boldsymbol{n} \leq 1000$).

```
Algoritmus mitCsinál(n, i):
    Ha i * i > n akkor
        visszatérít 0
    vége(ha)
    Ha i * i = n akkor
        visszatérít i
    vége(ha)
    Ha n MOD i = 0 akkor
        visszatérít i + n DIV i + mitCsinál(n, i + 1)
    különben
        visszatérít mitCsinál(n, i + 1)
    vége(ha)
Vége(algoritmus)
```

Az alábbi állítások közül melyek igazak a `mitCsinál(n, 2)` hívás esetén?

A. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi osztói összegének kétszeresét.
B. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi osztóinak összegét.
C. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi és nem valódi osztóinak összegét.
D. Az algoritmus megvizsgálja, hogy az ***n*** szám négyzetszám-e. Amennyiben az ***n*** négyzetszám, visszatéríti az ***n*** négyzetgyökét, különben 0-át.

20. Legyen a `mitCsinál(T, n, e)` algoritmus, amelynek bemeneti paraméterei az ***n*** természetes számot tartalmazó, növekvően rendezett ***T*** sorozat (***T***[1], ***T***[2], …, ***T***[***n***]), valamint az ***n*** és ***e*** természetes számok ($1 \leq \boldsymbol{n}, \boldsymbol{e} \leq 10000$).

```
Algoritmus mitCsinál(T, n, e):
    Ha e MOD 2 = 0 akkor
        a ← 1
        b ← n
        Amíg a ≤ b végezd el
            m ← (a + b) DIV 2
            Ha e < T[m] akkor
                b ← m - 1
            különben
                Ha e > T[m] akkor
                    a ← m + 1
                különben
                    visszatérít igaz
                vége(ha)
            vége(ha)
        vége(amíg)
        visszatérít hamis
    különben
        c ← 1
        Amíg c ≤ n végezd el
            Ha e = T[c] akkor
                visszatérít igaz
            vége(ha)
            c ← c + 1
        vége(amíg)
        visszatérít hamis
    vége(ha)
Vége(algoritmus)
```

A következő állítások közül melyek igazak?

A. Az algoritmus nem ellenőrzi, hogy az ***e*** szám a ***T*** sorozat páros szám értékű pozícióján található.
B. Az algoritmus ellenőrzi, hogy az ***e*** szám megtalálható-e a ***T*** sorozatban, és ha az ***e*** szám páratlan, akkor a felhasznált kereső algoritmus a bináris keresés.
C. Az algoritmus ellenőrzi, hogy az ***e*** szám megtalálható-e a ***T*** sorozatban, és ha az ***e*** szám páros, akkor a felhasznált kereső algoritmus a bináris keresés.
D. Az algoritmus csak akkor ellenőrzi, hogy az ***e*** szám megtalálható-e a ***T*** sorozatban, ha az ***e*** szám páratlan.

21. Egyenlő oldalú háromszögeket szeretnénk rajzolni * (csillag) és . (pont) karakterek segítségével. Például, az alábbi háromszögnek az oldala ***n*** = 5 csillagból áll. Ahhoz, hogy a háromszöget kirajzoljuk, 12 csillagra és 23 pontra volt szükség.

```
....*
...*.*
..*...*
.*.....*
*.*.*.*.*
```

A következő állítások közül melyek igazak?

A. Ha ***n*** = 2, pontosan 3 csillagra és 4 pontra van szükség.
B. Ha ***n*** = 7, pontosan 18 csillagra és 52 pontra van szükség.
C. Ha ***n*** = 7, pontosan 18 csillagra és 48 pontra van szükség.
D. Ha ***n*** = 15, pontosan 42 csillagra és 288 pontra van szükség.

22. Azt mondjuk, hogy egy ***n*** karakterből álló sorozat antipalindrom, ha minden karakterpár, amely a szélektől egyenlő távolságra elhelyezkedő két karakterből áll, páronként különbözik (kivételt képez a középső, abban az esetben, ha ***n*** páratlan). Például, *asdfg* és *xlxe* antipalindromok, de *asdsg* nem antipalindrom.

Legyen az `antipalindrom(s, bal, jobb)` algoritmus, amelynek bemeneti paramétere az ***n*** (1 ≤ ***n*** ≤ 10000) karakterből álló ***s*** sorozat (***s***[1], ***s***[2], …, ***s***[***n***]) és a ***bal*** és ***jobb*** természetes számok.

A következő algoritmusok közül melyek fognak az `antipalindrom(s, 1, n)` hívásra akkor és csakis akkor *igaz* értéket visszatéríteni, ha az ***s*** sorozat antipalindrom?

A.
```
Algoritmus antipalindrom(s, bal, jobb):
    Ha bal = jobb akkor
        visszatérít igaz
    különben
        első ← s[bal]
        utolsó ← s[jobb]
        Ha első = utolsó akkor
            visszatérít hamis
        különben
            visszatérít antipalindrom(s, bal + 1, jobb - 1)
        vége(ha)
    vége(ha)
Vége(algoritmus)
```

B.

```
Algoritmus antipalindrom(s, bal, jobb):
    Ha bal ≥ jobb akkor
        visszatérít igaz
    vége(ha)
    első ← s[bal]
    utolsó ← s[jobb]
    Ha első = utolsó akkor
        visszatérít hamis
    különben
        visszatérít antipalindrom(s, bal + 1, jobb - 1)
    vége(ha)
Vége(algoritmus)
```

C.

```
Algoritmus antipalindrom(s, bal, jobb):
    Ha bal > jobb akkor
        visszatérít igaz
    különben
        első ← s[bal]
        utolsó ← s[jobb]
        Ha első ≠ utolsó akkor
            visszatérít hamis
        különben
            visszatérít antipalindrom(s, bal + 1, jobb - 1)
        vége(ha)
    vége(ha)
Vége(algoritmus)
```

D.

```
Algoritmus antipalindrom(s, bal, jobb):
    Ha bal > jobb akkor
        visszatérít igaz
    vége(ha)
    első ← s[bal]
    utolsó ← s[jobb]
    Ha első ≠ utolsó akkor
        visszatérít igaz
    vége(ha)
    visszatérít antipalindrom(s, bal + 1, jobb - 1)
Vége(algoritmus)
```

23. Legyen az `ordo(n, a)` algoritmus, amelynek bemeneti paraméterei az ***n*** természetes szám ($1 \leq n \leq 10000$) és a **2*n*** elemű ***a*** sorozat, amelynek elemei természetes számok (***a***[1], ***a***[2], …, ***a***[2***n***]).

Tudjuk, hogy az ***a*** sorozat páros értékű elemeinek darabszáma egyenlő a páratlan értékű elemek darabszámával. Az alábbi algoritmusok közül melyek rendezik át az ***a*** sorozat elemeit úgy, hogy a páratlan értékű elemek páratlan indexű pozíciókon, a páros értékű elemek pedig páros indexű pozíciókon helyezkedjenek el?

A.

```
Algoritmus ordo(n, a):
    Minden i ← 1, 2 * n - 1 végezd el
        Ha a[i] MOD 2 ≠ i MOD 2 akkor
            Minden j ← i + 1, 2 * n végezd el
                Ha a[j] MOD 2 ≠ j MOD 2 akkor
                    a[i] ← a[i] + a[j]
                    a[j] ← a[i] - a[j]
                    a[i] ← a[i] - a[j]
                vége(ha)
            vége(minden)
        vége(ha)
    vége(minden)
Vége(algoritmus)
```

B.
```
Algoritmus ordo(n, a):
    Minden i ← 1, 2 * n - 1 végezd el
        Ha a[i] MOD 2 ≠ i MOD 2 akkor
            Minden j ← i + 1, 2 * n végezd el
                Ha (a[i] MOD 2 ≠ i MOD 2) ÉS
                   (a[j] MOD 2 ≠ j MOD 2) akkor
                    a[i] ← a[i] + a[j]
                    a[j] ← a[i] - a[j]
                    a[i] ← a[j] - a[i]
                vége(ha)
            vége(minden)
        vége(ha)
    vége(minden)
Vége(algoritmus)
```

C.
```
Algoritmus ordo(n, a):
    Minden i ← 1, 2 * n - 1 végezd el
        Ha a[i] MOD 2 ≠ i MOD 2 akkor
            Minden j ← i + 1, 2 * n végezd el
                Ha (a[i] MOD 2 ≠ i MOD 2) ÉS
                   (a[j] MOD 2 ≠ j MOD 2) ÉS
                   (a[i] MOD 2 ≠ a[j] MOD 2) akkor
                    a[i] ← a[i] + a[j]
                    a[j] ← a[i] - a[j]
                    a[i] ← a[i] - a[j]
                vége(ha)
            vége(minden)
        vége(ha)
    vége(minden)
Vége(algoritmus)
```

D.
```
Algoritmus ordo(n, a):
    Minden i ← 1, 2 * n - 1 végezd el
        Minden j ← i + 1, 2 * n végezd el
            Ha (a[j] MOD 2 = 0) ÉS
               ((a[j] MOD 2 ≠ 0) VAGY (a[j] MOD 2 ≠ 0)) ÉS
               (a[j] MOD 2 = 0) akkor
                a[i] ← a[i] + a[j]
                a[j] ← a[i] - a[j]
                a[i] ← a[i] - a[j]
            vége(ha)
        vége(minden)
    vége(minden)
Vége(algoritmus)
```

24. Fel szeretnénk osztani egy $\boldsymbol{n}$ elemű sorozatot ($1 \leq \boldsymbol{n} \leq 1000$) $\boldsymbol{k}$ darab ($1 \leq \boldsymbol{k} \leq \boldsymbol{n}$) egymást követő, azonos hosszúságú tömbszakaszra (a sorozat minden eleme pontosan egy tömbszakaszhoz tartozik). Ha $\boldsymbol{n}$ nem osztható $\boldsymbol{k}$-val, bármely két tömbszakasz hosszúságának különbsége legtöbb 1 lehet.

Adott négy lehetséges mód, amelyek generálják minden $\boldsymbol{j}$-edik ($1 \leq \boldsymbol{j} \leq \boldsymbol{k}$) tömbszakasz első elemének indexét. Ha a sorozat elemeit 1-től kezdve számozzuk, a négy változatból melyek elégítik ki a fenti követelményeket?

A. `((j * n - 1) DIV k) - 1`
B. `((j - 1) * n) DIV k + 1`
C. `(j - 1) * (n DIV k)`
D. `((j - 1) * n + k) DIV k`

25. Legyen $\boldsymbol{b_n b_{n-1} ... b_0}$ a $\boldsymbol{B}$ természetes szám alakja a kettes számrendszerben ($2021 \le \boldsymbol{B} \le 2021^{2021}$).

Az alábbi állítások közül melyek igazak?

A. Ha a $\boldsymbol{b_0} - \boldsymbol{b_1} + \boldsymbol{b_2} - \boldsymbol{b_3} + ... + (-1)^n * \boldsymbol{b_n}$ kifejezés értéke egyenlő 0-val, akkor $\boldsymbol{B}$ osztható 3-mal
B. Ha a $\boldsymbol{b_0} - \boldsymbol{b_1} + \boldsymbol{b_2} - \boldsymbol{b_3} + ... + (-1)^n * \boldsymbol{b_n}$ kifejezés értéke osztható 3-mal, akkor $\boldsymbol{B}$ osztható 3-mal
C. $\boldsymbol{B}$ osztható 3-mal, ha a bináris alak számjegyeinek összege osztható 3-mal, de nem osztható 9-cel
D. Ha $\boldsymbol{B}$ osztható 3-mal, akkor a $\boldsymbol{b_0} - \boldsymbol{b_1} + \boldsymbol{b_2} - \boldsymbol{b_3} + ... + (-1)^n * \boldsymbol{b_n}$ kifejezés értéke osztható 3-mal

26. Legyen a `prefix(n)` algoritmus, amely adott $\boldsymbol{n}$ ($9 < \boldsymbol{n} < 10^{20} - 1$) természetes számra megkeresi az $\boldsymbol{n}$ szám leghosszabb előszeletét (prefixét), amely megtalálható a szám belsejében is (kivéve az első és utolsó számjegyét). Az algoritmus visszatéríti a megfelelő előszelet hosszát.
Példa:

Ha $\boldsymbol{n}$ = 12133121, az előszelet 12, amelynek a hossza 2.
Ha $\boldsymbol{n}$ = 34534536, az előszelet 3453, amelynek a hossza 4.
Ha $\boldsymbol{n}$ = 1223, nem létezik megfelelő előszelet (a hosszát 0-nak tekintjük).

A sorozat elemeit 1-től kezdődően számozzuk. A következő algoritmusok közül melyek helyes változatai a `prefix(n)` algoritmusnak?

A.

```
Algoritmus prefix(n):
  szám ← n
  c ← 0
  p ← 1
  Amíg szám > 0 végezd el
    c ← c + 1
    szám ← szám DIV 10
    p ← p * 10
  vége(amíg)

  f1 ← 100
  f2 ← p DIV 100
  k ← 1
  ok ← 0
  Amíg ok = 0 végezd el
    n1 ← n DIV f1
    f3 ← 10
    Minden i ← 1 ,k végezd el
      n2 ← (n DIV f3) MOD f2
      Ha n1 = n2 akkor
        ok ← 1
        visszatérít c - k - 1
      vége(ha)
      f3 ← f3 * 10
    vége(minden)
    f1 ← f1 * 10
    f2 ← f2 DIV 10
    k ← k + 1
  vége(amíg)
  visszatérít -1
Vége(algoritmus)
```

B.

```
Algoritmus prefix(n):
    c ← [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
    szám ← n
    p ← 0
    Amíg szám > 0 végezd el
        c[p + 1] ← szám MOD 10
        szám ← szám DIV 10
        p ← p + 1
    vége(amíg)
    Minden i ← 1, p - 2 végezd el
        Minden j ← p – 1, i + 1, -1 végezd el
            ok ← 1
            Minden k ← 0, i - 1 végezd el
                Ha c[p - 1 - k] ≠ c[j - k] akkor
                  ok ← 0
                vége(ha)
            vége(minden)
            Ha ok = 1 akkor
                visszatérít i
            vége(ha)
        vége(minden)
    vége(minden)
    visszatérít -1
Vége(algoritmus)
```

C.

```
Algoritmus prefix(n):
    c ← [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
    szám ← n
    p ← 0
    Amíg szám > 0 végezd el
        c[p + 1] ← szám MOD 10
        szám ← szám DIV 10
        p ← p + 1
    vége(amíg)
    Minden i ← p - 2, 1, -1 végezd el
        Minden j ← p – 1, i + 1, -1 végezd el
            ok ← 1
            Minden k ← 0, i - 1 végezd el
                Ha c[p - k] ≠ c[j - k] akkor
                    ok ← 0
                vége(ha)
            vége(minden)
            Ha ok = 1 akkor
                visszatérít j
            vége(ha)
        vége(minden)
    vége(minden)
    visszatérít 0
Vége(algoritmus)
```

D.

```
Algoritmus prefix(n):
    szám ← n
    c ← 0
    p ← 1
    Amíg szám > 0 végezd el
        c ← c + 1
        szám ← szám DIV 10
        p ← p * 10
    vége(amíg)
    f1 ← p DIV 10
    f2 ← 10
    k ← c - 2
    ok ← -1
    Minden t ← 1, c - 2 végezd el
        n1 ← n DIV f1
        f3 ← 10
        Minden i ← 1, k végezd el
            n2 ← (n DIV f3) MOD f2
            Ha n1 = n2 akkor
                Ha ok < c - k - 1 akkor
                    ok ← c - k - 1
                vége(ha)
            vége(ha)
            f3 ← f3 * 10
        vége(minden)
        f1 ← f1 DIV 10
        f2 ← f2 * 10
        k ← k - 1
    vége(minden)
    Ha ok < 0 akkor:
        visszatérít 0
    vége(ha)
    visszatérít ok
Vége(algoritmus)
```

27. Legyen az alábbi táblázat, amelynek 16 cellája van (4 sora, amelyek 1-től 4-ig vannak számozva és 4 oszlopa A, B, C, D-vel jelölve). Egyes cellák konstansokat tárolnak (pl. a B3-as cella), mások, ahol a tartalom „=” jellel kezdődik, egy kéttagú aritmetikai kifejezés eredményét tartalmazzák. Minden tag vagy egy konstans, vagy, ha a tag a „$” jellel kezdődik, akkor egy hivatkozás egy másik cellában található értékre. Például, az A4 cella egy kivonás aritmetikai művelet eredményét tartalmazza, vagyis az 5 konstans és az A2 cellában található érték különbségét. Adott ***i*** cellára jelöljük ***X***(***i***)-vel a cellák azon minimális darabszámát (beleszámítva azokat is, amelyek konstansokat tartalmaznak) amelyeknek értékeit ismernünk kell mielőtt az ***i*** cellában levő kifejezés értékét kiszámolhatnánk. Hasonlóan, jelöljük ***Y***(***i***)-vel a cellák azon maximális darabszámát (beleszámítva azokat is, amelyek konstansokat tartalmaznak, de kivéve az ***i*** cellát) amelyeknek értékeit ki tudjuk számítani, anélkül, hogy ismernénk az ***i*** cellában lévő kifejezés értékét.
Az alábbi állítások közül melyek igazak az ***X***(A2) és az ***Y***(A2) értékeit illetően?

	A	B	C	D
1	= $B4 – $C1	=$B3 + $D3	3	= $A4 * $C3
2	= $B1 + $B2	= $D3 + 11	= $D3 + $D2	2
3	= $B1 – $D3	11	= $D4 * $D4	= $D2 + 2
4	= 5 - $A2	= $C1 * $C1	= $A3 / 2	=15 / 3

A. ***X***(A2) = 2 és ***Y***(A2) = 1
B. ***X***(A2) = 5 és ***Y***(A2) = 13
C. ***X***(A2) = 6 és ***Y***(A2) = 4
D. ***X***(A2) = ***X***(C4) és ***Y***(A2) = ***Y***(B2) + 1

28. Legyen az `egyszerűsít(számláló, nevező)` algoritmus, amely előállítja a ***segéd*1 / *segéd*2** irreducibilis törtet, amelyre érvényes a következő tulajdonság: ***segéd*1 / *segéd*2** = ***számláló* / *nevező*** (ahol ***segéd*1**, ***segéd*2**, ***számláló***, ***nevező*** természetes számok és ***nevező***, ***segéd*2** ≠ 0).

```
Algoritmus egyszerűsít(számláló, nevező):
    osztó ← függvény(számláló, nevező)
    segéd1 ← számláló DIV osztó
    segéd2 ← nevező DIV osztó
Vége(algoritmus)
```

Az alábbi `függvény(a, b)` algoritmusok közül melyek helyesek?

A.
```
Algoritmus függvény(a, b):
    osztó ← 1
    Amíg igaz végezd el
        Ha (a MOD osztó = 0) ÉS (b MOD osztó = 0) akkor
            visszatérít osztó
        vége(ha)
        osztó ← osztó + 1
    vége(amíg)
Vége(algoritmus)
```

B.
```
Algoritmus függvény(a, b):
    Amíg b ≠ 0 végezd el
        c ← a MOD b
        a ← b
        b ← c
    vége(amíg)
    visszatérít a
Vége(algoritmus)
```

C.
```
Algoritmus függvény(a, b):
    Amíg a ≠ b végezd el
        Ha a > b akkor
            a ← a - b
        különben
            b ← b - a
        vége(ha)
    vége(amíg)
    visszatérít a
Vége(algoritmus)
```

D.
```
Algoritmus függvény(a, b):
    osztó ← a
    Amíg ((a MOD osztó ≠ 0) VAGY (b MOD osztó ≠ 0)) végezd el
        osztó ← osztó - 1
    vége(amíg)
    visszatérít osztó
Vége(algoritmus)
```

29. Legyen a `fibonacci(n)` rekurzív algoritmus, ahol ***n*** természetes szám ($1 \leq n \leq 100$). Határozzuk meg, hogy hányszor írja ki a algoritmus az "`Itt`" szöveget, ha az algoritmust a `fibonacci(n)` utasítással hívjuk meg, majd ez rekurzívan többször meghívja önmagát.

```
Algoritmus fibonacci(n):
    Ha n ≤ 1 akkor
        visszatérít 1
    különben
        Ki: "Itt"
        visszatérít fibonacci(n - 1) + fibonacci(n - 2)
    vége(ha)
Vége(algoritmus)
```

A. `fibonacci(n)`-szer.
B. `fibonacci(n – 1)`-szer.
C. `(fibonacci(n) – 1)`-szer.
D. `(fibonacci(n) – fibonacci(n – 1))`-szer.

30. Adott a következő kifejezés: $E(x) = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3 + a_5 * x^5$, ahol a_0, a_1, a_2, a_3, a_5 és x nullától különböző valós számok. Az $E(x)$ kifejezés értékének kiszámítása érdekében elvégzendő szorzások minimális darabszáma:

A. 4
B. 5
C. 7
D. 11