BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA–INFORMATIKA KAR

Felvételi verseny – 2022 július 19.

Informatika írásbeli

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

1. Adott a `mitCsinál(a, b)` algoritmus, ahol ***a*** és ***b*** természetes számok (a hívás pillanatában $1 \le \boldsymbol{a}, \boldsymbol{b} \le 10000$).

```
Algorithm mitCsinál(a, b):
    While (a MOD 10 = b MOD 10) AND (a ≠ 0) AND (b ≠ 0) execute
        a ← a DIV 10
        b ← b DIV 10
    EndWhile
    If ((a = 0) AND (b = 0)) then
        return True
    else
        return False
    EndIf
EndAlgorithm
```

A `mitCsinál(a, b)` algoritmus akkor és csakis akkor térít vissza *True*-t, ha:

A. ***a*** és ***b*** számjegyeinek darabszáma azonos
B. ***a*** egyenlő ***b***-vel
C. ***a*** és ***b*** ugyanazokat a számjegyeket tartalmazza, de más sorrendben
D. ***a*** utolsó számjegye egyenlő ***b*** utolsó számjegyével

2. Adott az `f(a, n)` algoritmus, ahol ***n*** egy nullától különböző természetes szám ($2 \le \boldsymbol{n} \le 10000$) és ***a*** egy egész számokból álló ***n*** elemű vektor ($\boldsymbol{a}[1], \boldsymbol{a}[2], ..., \boldsymbol{a}[\boldsymbol{n}]$, $-100 \le \boldsymbol{a}[\boldsymbol{i}] \le 100$, $\boldsymbol{i} = 1, 2, ..., \boldsymbol{n}$). A ***b*** lokális változó egy vektor.

```
Algorithm f(a, n):
    i ← 2
    b[1] ← a[1]
    While i ≤ n execute
        b[i] ← b[i - 1] + a[i]
        i ← i + 1
    EndWhile
    return b[n]
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az algoritmus az ***a*** vektor összes elemének összegét téríti vissza.
B. Az algoritmus az ***a*** vektor utolsó két elemének összegét téríti vissza.
C. Az algoritmus az ***a*** vektor utolsó elemét téríti vissza.
D. Az algoritmus az ***a*** vektor utolsó $\boldsymbol{n} - 1$ elemének összegét téríti vissza.

3. Az alábbi algoritmusok közül melyik téríti vissza az ***n*** természetes szám prímtényezős felbontásában szereplő különböző prímszámok darabszámát (a hívás pillanatában $5 < n < 10^5$).

A.

```
// A prime vektor hossza n
// prime[i] értéke True, ha az i szám
// prímszám és False különben
Algorithm prímtényezőkSzáma_A(n, prime):
    d ← 2
    db ← 0
    p ← 0
    While n > 0 execute
        While n MOD d = 0 execute
            p ← p + 1
            n ← n DIV d
        EndWhile
        If p ≠ 0 then
            db ← db + 1
        EndIf
        d ← d + 1
        While prime[d] = False execute
            d ← d + 1
        EndWhile
        p ← 0
    EndWhile
    return db
EndAlgorithm
```

B.

```
Algorithm prímtényezőkSzáma_B(n):
    d ← 2
    db ← 0
    While n > 1 execute
        p ← 0
        While n MOD d = 0 execute
            p ← p + 1
            n ← n DIV d
        EndWhile
        If p > 0 then
            db ← db + 1
        EndIf
        If d = 2 then
            d ← d + 1
        else
            d ← d + 2
        EndIf
    EndWhile
    return db
EndAlgorithm
```

C.

```
Algorithm prímtényezőkSzáma_C(n):
    db ← 0
    For d ← 2, n execute
        If n MOD d = 0 then
            db ← db + 1
        EndIf
        While n MOD d = 0 execute
            n ← n DIV d
        EndWhile
    EndFor
    return db
EndAlgorithm
```

D.

```
Algorithm prímtényezőkSzáma_D(n):
    db ← 0
    d ← 2
    While d * d ≤ n execute
        If n MOD d = 0 then
            db ← db + 1
        EndIf
        While n MOD d = 0 execute
            n ← n DIV d
        EndWhile
        d ← d + 1
    EndWhile
    return db
EndAlgorithm
```

4. Adott a `mitCsinál(n, m)` algoritmus, ahol ***n*** természetes szám ($0 \leq n \leq 1000$). Az ***n*** szám utolsó számjegye 0-tól különbözik.

```
Algorithm mitCsinál(n, m):
    If n = 0 then
        return m
    else
        return mitCsinál(n DIV 10, m * 10 + n MOD 10)
    EndIf
EndAlgorithm
```

Mi lesz az eredménye a `mitCsinál(n, 0)` hívásnak?

A. 0 (függetlenül ***n*** értékétől)
B. ***n*** (függetlenül ***n*** értékétől)
C. Az ***n*** szám számjegyeinek összege
D. Az ***n*** szám tükrözöttje

5. Adott az `f(x, n)` algoritmus, ahol ***n*** természetes szám ($2 \leq \boldsymbol{n} \leq 10000$) és ***x*** egy természetes számokból álló ***n*** elemű sorozat (***x***[1], ***x***[2], ..., ***x***[***n***], $1 \leq \boldsymbol{x}[\boldsymbol{i}] \leq 10000$, ***i*** = 1, 2, ..., ***n***).

```
Algorithm f(x, n):
    For i = 1, n – 1 execute
        If x[i] = x[i + 1] then
            return False
        EndIf
    EndFor
    return True
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az algoritmus *False*-t térít vissza, ha az ***x*** sorozat két tetszőleges eleme különbözik.
B. Az algoritmus *False*-t térít vissza, ha az ***x*** sorozat két tetszőleges eleme egyenlő.
C. Az algoritmus *False*-t térít vissza, ha az ***x*** sorozat két egymást követő eleme egyenlő.
D. Az algoritmus *False*-t térít vissza, ha az ***x*** sorozat első két eleme egyenlő.

6. Adott az `f(x, n)` algoritmus, ahol ***x*** és ***n*** természetes számok ($0 \leq \boldsymbol{n} \leq 10000$, $0 < \boldsymbol{x} \leq 10000$).

```
1. Algorithm f(x, n):
2.     If n = 0 then
3.         return 1
4.     EndIf
5.     m ← n DIV 2
6.     p ← f(x, m)
7.     If n MOD 2 = 0 then
8.         return p * p
9.     EndIf
10.    return x * p * p
11.EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az algoritmus az ***x*** az ***n*****.** hatványon értékét téríti vissza.
B. Ha a 7. sorban, `n MOD 2` helyett `m MOD 2` lenne, akkor az algoritmus az ***x*** az ***n*****.** hatványon értékét térítené vissza.
C. A 6. sorban található újrahívás következtében a 7., 8., 9., 10. sorban található utasításokat az algoritmus egyszer sem hajtja végre.
D. Az algoritmus 1-et térít vissza, ha ***n*** páros szám vagy ***x*** értékét, ha ***n*** páratlan szám.

7. Feltételezve, hogy minden szorzási és osztási művelet végrehajtásának ideje konstans, mit mondhatunk a **6.** tételben szereplő algoritmus időbonyolultságáról?

A. Az időbonyolultság függ az ***x*** és ***n*** paraméterektől.
B. Az időbonyolultság nem függ az ***x*** paramétertől.
C. Az időbonyolultság $O(\log \log \boldsymbol{n})$.
D. Az időbonyolultság logaritmikus az ***n*** paraméter függvényében ($O(\log \boldsymbol{n})$).

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

```
Algorithm kiír(n):
    If n ≤ 4000 then
        Write n, " "
        kiír(2 * n)
        Write n, " "
    EndIf
EndAlgorithm
```

Mit fog kiírni a program a `kiír(1000)` hívás következtében?

A. 1000 2000 4000
B. 1000 2000 4000 4000 2000 1000
C. 1000 2000 4000 2000 1000
D. 1000 2000 2000 1000

9. Melyek lehetnek egy olyan vektor elemei, amelyre, ha alkalmazzuk a bináris keresés algoritmusát és a 36-os értéket keressük, az összehasonlítások egymás után rendre a 12, 24, 36 értékekkel történnek?

A. [2, 4, 7, 12, 24, 36, 50]
B. [2, 4, 8, 9, 12, 16, 20, 24, 36, 67]
C. [4, 8, 9, 12, 16, 24, 36]
D. [12, 24, 36, 42, 54, 66]

10. A következő kifejezések közül melyek ekvivalensek az `x MOD y` kifejezéssel minden $\boldsymbol{x}$ és $\boldsymbol{y}$ ($0 < \boldsymbol{x}, \boldsymbol{y} \leq 10000$) szigorúan pozitív természetes szám esetében?

A. `x DIV y`
B. `x – (y * (x DIV y))`
C. `x – (x * (x DIV y))`
D. `x DIV y + y DIV x`

11. Legyen az $\boldsymbol{n}$ változó, amely egy természetes számot tárol. Az alábbi kifejezések közül melyeknek van akkor és csakis akkor *True* értékük, ha $\boldsymbol{n}$ osztható 2-vel és 3-mal?

A. `(n DIV 2 = 0) OR (n DIV 3 ≠ 0)`
B. `(n MOD 3 = 2) OR (n MOD 2 = 3)`
C. `(n MOD 2 ≠ 1) AND (n MOD 3 = 0)`
D. `(n MOD 2 = 0) AND (n MOD 3 ≠ 1)`

12. Legyen az $\boldsymbol{n}$ változó, amely egy természetes számot tárol. Az alábbi kifejezések közül melyeknek van akkor és csakis akkor *True* értékük, ha $\boldsymbol{n}$ osztható 2-vel és 3-mal?

A. `(n MOD 2) - (n MOD 3) = 0`
B. `(n MOD 2) - (n MOD 3) < 0`
C. `(n MOD 2) + (n MOD 3) > 0`
D. `(n MOD 2) + (n MOD 3) = 0`

13. Adott az `f(n)` algoritmus, ahol $\boldsymbol{n}$ természetes szám ($1 \leq \boldsymbol{n} \leq 100$). A”/” műveleti jel valós osztást jelent (például 3 / 2 = 1,5). Adjátok meg az algoritmus hatását.

```
Algorithm f(n):
    s ← 0; p ← 1;
    For i ← 1, n execute
        s ← s + i
        p ← p * (1 / s)
    EndFor
    return p
EndAlgorithm
```

A. Kiszámítja az $1 / 1 * 1 / 2 * 1 / 3 * ... * 1 / \boldsymbol{n}$ kifejezés értékét.
B. Kiszámítja az $1 / 1 * 1 /(1 * 2) * 1 / (1 * 2 *3) * ... * 1 /(1 * 2 * 3 * … * \boldsymbol{n})$ kifejezés értékét.
C. Kiszámítja az $1 / 1 * 1 / (1 + 2) * 1 / (1 + 2 + 3) * ... * 1 / (1 + 2 + 3 + … + \boldsymbol{n})$ kifejezés értékét.
D. Kiszámítja az $1 / 1 + 1 / (1 * 2) + 1 / (1 * 2 * 3) + ... + 1 / (1 * 2 * 3 * … * \boldsymbol{n})$ kifejezés értékét.

14. Adott a `feldolgozás(s1, hossz1, s2, hossz2)` algoritmus, ahol ***s*1** és ***s*2** két karakterlánc, amelyeknek a hossza ***hossz*1** és ***hossz*2** ($1 \leq \boldsymbol{hossz1}, \boldsymbol{hossz2} \leq 1000$). A két karakterlánc csak olyan karaktereket tárol, amelyeknek ASCII kódja az [1, 125] intervallumhoz tartozik. Az ***x*** lokális változó vektor. Az `ascii(s, i)` algoritmus az ***s*** karakterlánc ***i.*** karakterének ASCII kódját téríti vissza.

```
Algorithm feldolgozás(s1, hossz1, s2, hossz2):
    For i = 1, 125 execute
        x[i] ← 0
    EndFor
    For i = 1, hossz1 execute
        x[ascii(s1, i)] ← x[ascii(s1, i)] + 1
    EndFor
    For i = 1, hossz2 execute
        x[ascii(s2, i)] ← x[ascii(s2, i)] - 1
    EndFor
    ok ← True
    For i = 1, 125 execute
        If x[i] ≠ 0 then
            ok ← False
        EndIf
    EndFor
    return ok
EndAlgorithm
```

Adjátok meg az algoritmus hatását.

A. Az algoritmus *True*-t térít vissza, ha az ***s*1** és ***s*2** karakterlánc hossza azonos, különben *False*-t.
B. Az algoritmus *True*-t térít vissza, ha az ***s*1** és ***s*2** karakterláncok ugyanazokat a karaktereket tartalmazzák, minden karakter esetében ugyanazzal a gyakorisággal, különben *False*-t.
C. Az algoritmus *True*-t térít vissza, ha egyenként mindkét ***s*1** és ***s*2** karakterláncban előfordul minden karakter, amelyeknek ASCII kódja az [1, 125] intervallumhoz tartozik, különben *False*-t.
D. Az algoritmus *True*-t térít vissza, ha az ***s*1** és ***s*2** karakterláncok különböző karakterekből állnak, különben *False*-t.

15. Mi lesz az 100101100111 bináris szám értéke a 10-es számrendszerben?

A. 2407
B. 2408
C. 1203
D. Az A., B., C. válaszok egyike sem

16. Adott az ***n*** elemű, természetes számokat tároló ***a*** vektor (***a***[1], ***a***[2], …, ***a***[***n***]), valamint az ***n*** és ***x*** természetes számok ($1 \leq \boldsymbol{n} \leq 10000$). A következő pszeudokód-részletek közül melyik írja ki azt a legkisebb pozíciót, ahol az ***a*** vektorban megtalálható az ***x*** érték, vagy -1-et ha ***x*** nem található meg az ***a*** vektorban?

A.

```
i ← 1
While (i ≤ n) AND (a[i] = x) execute
    i ← i + 1
EndWhile
If i ≤ n then
    Write i
else
    Write -1
EndIf
```

B.

```
i ← 1
While (i ≤ n) AND (a[i] ≠ x) execute
    i ← i + 1
EndWhile
If i = n + 1 then
    Write i
else
    Write -1
EndIf
```

C.
```
i ← 1
While (i ≤ n) AND (a[i] = x) execute
    i ← i + 1
EndWhile
If i = n + 1 then
    Write i
else
    Write -1
EndIf
```

D.
```
i ← 1
While (i ≤ n) AND (a[i] ≠ x) execute
    i ← i + 1
EndWhile
If i ≤ n then
    Write i
else
    Write -1
EndIf
```

17. Adott az `f(x)` algoritmus, ahol ***x*** egész szám:

```
Algorithm f(x):
    If x = 0 then
        return 0
    else
        If x MOD 3 = 0 then
            return f(x DIV 10) + 1
        else
            return f(x DIV 10)
        EndIf
    EndIf
EndAlgorithm
```

Az ***x*** változó mely értékére térít vissza az algoritmus 4-et?

A. 13369 B. 21369 C. 4 D. 1233

18. Adott az `f(n, i, j)` algoritmus, ahol ***n***, ***i*** és ***j*** természetes számok (az eredeti hívás pillanatában $1 \le \boldsymbol{n}, \boldsymbol{i}, \boldsymbol{j} \le 10000$).

```
Algorithm f(n, i, j):
    If i > j then
        Write '*'
    else
        If n MOD i = 0 then
            f(n, i - 1, j)
        else
            If n DIV i ≠ j then
                f(n, i + 1, j - 1)
                Write '0'
            else
                f(n, i + 2, j - 2)
                Write '#'
            EndIf
        EndIf
    EndIf
EndAlgorithm
```

Mit fog kiírni a program az `f(15, 3, 10)` hívás következtében?

A. *000000
B. *0#000
C. *0#0000
D. *0000000

19. Adott a `mitCsinál(n, x)` algoritmus, ahol $\boldsymbol{n}$ természetes szám ($1 \leq \boldsymbol{n} \leq 100$) és $\boldsymbol{x}$ egy $\boldsymbol{n}$ elemű, természetes számokat tároló vektor ($\boldsymbol{x}[1], \boldsymbol{x}[2], \ldots, \boldsymbol{x}[\boldsymbol{n}]$).

```
Algorithm mitCsinál(n, x):
    For i = 1, n execute
        c ← x[i]
        x[i] ← x[n - i + 1]
        x[n - i + 1] ← c
    EndFor
EndAlgorithm
```

Mit fog tárolni az $\boldsymbol{x}$ vektor az algoritmus végrehajtása után, ha $\boldsymbol{n} = 6$ és $\boldsymbol{x} = [5, 3, 2, 1, 1, 1]$?

A. [1, 1, 2, 1, 3, 5] B. [1, 1, 1, 2, 3, 5] C. [5, 3, 2, 1, 1, 1]
D. egyik előző változat sem helyes.

20. Adott a `what(n)` algoritmus, ahol $\boldsymbol{n}$ természetes szám (az eredeti hívás pillanatában $1 \leq \boldsymbol{n} \leq 1000$).

```
Algorithm what(n):
    If n = 0 then
        return True
    EndIf
    If (n MOD 10 = 3) OR (n MOD 10 = 7) then
        return what(n DIV 10)
    else
        return False
    EndIf
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az algoritmus akkor és csakis akkor térít vissza *True*-t, ha $\boldsymbol{n}$ csak 3-as számjegyekből áll, vagy ha $\boldsymbol{n}$ csak 7-es számjegyekből áll
B. Az algoritmus *False*-t térít vissza, ha az $\boldsymbol{n}$ számjegyei között található legalább egy páros számjegy
C. Az algoritmus akkor és csakis akkor térít vissza *False*-t, ha $\boldsymbol{n}$ számjegyei között található legalább egy $\boldsymbol{c}$ értékű számjegy, ahol $\boldsymbol{c} \neq 3$ és $\boldsymbol{c} \neq 7$
D. Az algoritmus akkor és csakis akkor térít vissza *True*-t, ha $\boldsymbol{n}$ számjegyei között nem fordul elő egy sem a {0, 1, 2, 4, 5, 6, 8, 9} halmazból

21. Adott a `számol(x, n)` algoritmus, ahol $\boldsymbol{x}$ és $\boldsymbol{n}$ természetes számok ($1 \leq \boldsymbol{x} \leq 10000$, $1 \leq \boldsymbol{n} \leq 10000$), és $\boldsymbol{x} \leq \boldsymbol{n}$.

```
Algorithm számol(x, n):
    b ← 1
    For i ← 1, n – x execute
        b ← b * i
    EndFor
    a ← b
    For i ← n - x + 1, n execute
        a ← a * i
    EndFor
    return a DIV b
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha $\boldsymbol{x} = 2$ és $\boldsymbol{n} = 5$, az algoritmus 10-et térít vissza.
B. Az algoritmus az $\{1, 2, \ldots, \boldsymbol{n}\}$ halmaz $\boldsymbol{x}$ elemű részhalmazainak a darabszámát téríti vissza.
C. Az algoritmus $\boldsymbol{n}$ elem $\boldsymbol{x}$. rendű variációinak darabszámát téríti vissza.
D. Az algoritmus $\boldsymbol{n}$ elem $\boldsymbol{x}$. rendű kombinációinak darabszámát téríti vissza.

22. Egy farmon tyúkokat és nyulakat tenyésztenek. Minden tyúknak két lába van, minden nyúlnak négy. A farmon található állatoknak összesen $\boldsymbol{n}$ feje és $\boldsymbol{m}$ lába van ($0 \leq \boldsymbol{n}, \boldsymbol{m} \leq 10^4$). A következő algoritmusok közül melyik térít vissza *True*-t, és ír ki minden olyan lehetséges számpárt, amely a farmon élő tyúkok és nyulak darabszámát jelölheti, vagy térít vissza *False*-t, ha nincs megoldás?

A.
```
Algorithm ferma_A(n, m):
    found = False
    For i ← 0, n execute
        j ← n - i
        If 2 * i + 4 * j = m then
            found ← True
            Write i, ' ', j
            Write newline
        EndIf
    EndFor
    return found
EndAlgorithm
```

B.
```
Algorithm ferma_B(n, m):
    found ← False
    For i ← 0, n execute
        For j ← 0, n execute
            If 2 * i + 4 * j = m AND
                                  i + j = n then
                found ← True
                Write i, ' ', j
                Write newline
            EndIf
        EndFor
    EndFor
    return found
EndAlgorithm
```

C.
```
Algorithm ferma_C(n, m):
    found ← False
    For i ← 0, n execute
        For j ← 0, n - i execute
            If 2 * i + 4 * j = m AND
                       i + j = n then
                found ← True
                Write i, ' ', j
                Write newline
            EndIf
        EndFor
    EndFor
    return found
EndAlgorithm
```

D.
```
Algorithm ferma_D(n, m):
    found ← False
    For i ← 0, n execute
        For j ← 0, i execute
            If 2 * i + 4 * j = m AND
                                  i + j = n then
                found ← True
                Write i, ' ', j
                Write newline
            EndIf
        EndFor
    EndFor
    return found
EndAlgorithm
```

23. Adott az $\boldsymbol{n}$ természetes szám, amely felírható három természetes szám szorzataként ($\boldsymbol{n} = \boldsymbol{a} * \boldsymbol{b} * \boldsymbol{c}$). A következő kifejezések közül melyiknek egyenlő az értéke $\boldsymbol{n}$-nek a $\boldsymbol{d}$ természetes számmal való osztási maradékával ($1 \leq \boldsymbol{n}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \leq 10000$)?

A. `(a MOD d) * b * c`
B. `((a MOD d) * (b MOD d) * (c MOD d)) MOD d`
C. `(a MOD d) * (b MOD d) * (c MOD d)`
D. `(a DIV d) * (b DIV d) * (c DIV d)`

24. Adott a `det(a, n, m)` algoritmus, ahol $\boldsymbol{a}$ egy $\boldsymbol{n}$ elemű, természetes számokból álló sorozat ($\boldsymbol{a}[1]$, $\boldsymbol{a}[2]$, ..., $\boldsymbol{a}[\boldsymbol{n}]$, ha $\boldsymbol{n} \geq 1$) vagy az üres sorozat, ha $\boldsymbol{n} = 0$. $\boldsymbol{n}$ és $\boldsymbol{m}$ természetes számok ($0 \leq \boldsymbol{n} \leq 100$, $0 \leq \boldsymbol{m} \leq 10^6$).

```
1.  Algorithm det(a, n, m):
2.      For i ← 1, n – 1 execute
3.          For j ← i + 1, n execute
4.              If a[i] > a[j] then
5.                  tmp ← a[i]
6.                  a[i] ← a[j]
7.                  a[j] ← tmp
8.              EndIf
9.          EndFor
10.     EndFor
11.     i ← 1
12.     j ← n
13.     b ← False
14.     While i < j execute
15.         If a[i] + a[j] = m then
16.             b ← True
17.         EndIf
18.         If a[i] + a[j] < m then
19.             i ← i + 1
20.         else
21.             j ← j - 1
22.         EndIf
23.     EndWhile
24.     return b
25. EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az algoritmus *True*-t térít vissza, ha az ***a*** sorozatban létezik egy számpár, amelynek összege ***m***.
B. Az algoritmus mindig *False*-t térít vissza.
C. Az algoritmus *False*-t térít vissza, ha ***n*** = 0.
D. Az 2., 3., …, 10. sorokban az algoritmus növekvő sorrendbe rendezi az ***a*** sorozatot.

25. Adott a `bűvös(n, a)` algoritmus, ahol ***a*** egy ***n*** elemű, természetes számokból álló vektor (***a***[1], ***a***[2], ..., ***a***[***n***], $1 \leq n \leq 10000$).

```
Algorithm bűvös(n, a):
    If n < 2 then
        return False
    EndIf
    For i ← 2, n execute
        If a[i – 1] = a[i] then
            return True
        EndIf
    EndFor
    return False
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. A `bűvös(5, [2, 5, 4, 5, 4])` hívás eredményeként az algoritmus *False*-t térít vissza.
B. Az algoritmus akkor és csakis akkor dönti el, hogy az ***a*** vektorban léteznek duplák, ha a vektor növekvően/csökkenően rendezett.
C. A `bűvös(9, [1, 2, 3, 4, 4, 5, 6, 7, 9])` hívás eredményeként az algoritmus *True*-t térít vissza.
D. A `bűvös(5, [9, 5, 5, 2, 4])` hívás eredményeként az algoritmus *True*-t térít vissza.

26. Adott az `f(n, a, b, c)` algoritmus, ahol ***n*** természetes szám ($n \leq 20$) és ***a***, ***b***, ***c*** három egész szám.

```
Algorithm f(n, a, b, c):
    If n = 0 then
        return 1
    else
        return f(n - 1, a * a, b + 1, c * 2) + f(n - 1, a - 1, b * b, c + 1) + 1
    EndIf
EndAlgorithm
```

Mi lesz a visszatérített eredmény az `f(n, 1, 1, 2)` hívás esetében

A. $2^{n+1} - 1$

B. n

C. $2^0 + 2^1 + 2^2 + ... + 2^n$

D. 2^{n+1}

27. Adottak az `f(n, p)` és a `g(n)` algoritmusok, ahol ***n*** és ***p*** eredetileg természetes számok (az eredeti hívás pillanatában $1 \leq n, p \leq 10^6$).

```
Algorithm g(n):
    If n < 2 then
        return False
    EndIf
    i ← 2
    While i * i ≤ n execute
        If n MOD i = 0 then
            return False
        EndIf
        i ← i + 1
    EndWhile
    return True
EndAlgorithm
```

```
Algorithm f(n, p):
    If n = 0 then
        return 1
    EndIf
    If n > 0 AND n ≥ p then
        c ← 0
        If g(p) = True then
            c ← c + f(n - p, p + 1)
        EndIf
        return c + f(n, p + 1)
    EndIf
    return 0
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. A `g(n)` algoritmus *True*-t térít vissza, ha ***n*** prímszám, különben *False*-t.

B. Az `f(n, 2)` hívás következtében az algoritmus visszatéríti, hogy hány különböző módon lehet növekvően rendezett különböző prímszámok legalább egy tagot tartalmazó összegeként felírni az ***n*** számot.

C. Az `f(n, 2)` hívás következtében az algoritmus az ***n*** szám prímosztóinak összegét téríti vissza.

D. Az `f(n, 1)` és az `f(n, 2)` hívások azonos eredményt térítenek vissza ***n*** bármely értékére.

28. Adott az `AlexB(value, n, k, p)` algoritmus, ahol ***value*** egy természetes számokból álló, ***n*** elemű sorozat (***value***[1], ***value***[2], ..., ***value***[***n***]), valamint ***n***, ***k*** és ***p*** természetes számok. Kezdetben a ***value*** sorozat ***n*** eleme 0-val egyenlő. A `kiírás(value, n)` algoritmus kiírja a ***value*** sorozatot egy sorba.

```
Algorithm AlexB(value, n, k, p):
    p ← p + 1
    value[k] ← p
    If p = n then
        kiírás(value, n)
    else
        For i ← 1, n execute
            If value[i] = 0 then
                AlexB(value, n, i, p)
            EndIf
        EndFor
    EndIf
    p ← p - 1
    value[k] ← 0
EndAlgorithm
```

Mit ír ki az algoritmus a 10. sorba, ha ***n*** = 5 és `AlexB(value, 5, 1, 0)` alakban hívjuk meg?

A. 1 5 2 3 4

B. 1 5 4 0 4

C. 5 5 5 5 5

D. 1 2 5 4 3

29. Adott az `f(n)` algoritmus, ahol ***n*** természetes szám (a hívás pillanatában $1 \le n \le 10000$).

```
Algorithm f(n):
    c ← 0
    If n ≠ 0 then
        c ← c + 1
        n ← n & (n - 1)       // bitenkénti and művelet
        While n ≠ 0 execute
            c ← c + 1
            n ← n & (n - 1)   // bitenkénti and művelet
        EndWhile
    EndIf
    return c
EndAlgoritm
```

Az & műveleti jel az AND bitenkénti műveletet jelöli; a művelet igazságtáblája a következő:

&	0	1
0	0	0
1	0	1

Példa:

2 & 7 felírva binárisan: 010 & 111 = 010, ami 10-es számrendszerben =2.

6 & 1 felírva binárisan: 110 & 001 = 000, ami 10-es számrendszerben =0.

Az alábbi állítások közül, melyek **NEM** igazak?

A. Ha ***n*** a 2-nek hatványa, akkor `f(n)` az 1 értéket téríti vissza.
B. Ha ***n*** >16 és ***n*** < 32, akkor az `f(n)` által visszatérített érték eleme az {2, 3, 4, 5} halmaznak.
C. Az algoritmus az ***n***-nél szigorúan kisebb páros számok darabszámát téríti vissza.
D. Az algoritmus az ***n***-nél kisebb páratlan számok darabszámát téríti vissza.

30. Adott a `számol(v, n)` algoritmus, ahol ***n*** egy nullától különböző természetes szám ($1 \le n \le 10000$) és ***v*** egy ***n*** egész számot tároló sorozat (***v***[1], ***v***[2], ..., ***v***[***n***]). A `return x, y` utasítás az (***x***, ***y***) értékpárt téríti vissza.

```
Algorithm számol(v, n):
    i ← n DIV 2 + 1
    j ← i + 1
    k ← i
    p ← j
    While j ≤ n execute
        While (j ≤ n) AND (v[i] = v[j]) execute
            j ← j + 1
        EndWhile
        If j - i > p - k then
            k ← i
            p ← j
        EndIf
        i ← j
        j ← j + 1
    EndWhile
    If j - i > p - k then
        k ← i
        p ← j
    EndIf
    return p - k, k
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha a sorozatnak csak egy eleme van, az algoritmus a 0, -1 értékeket téríti vissza
B. Ha $\boldsymbol{n}$ = 2 és a sorozat két eleme szimmetrikus a 0-ra nézve (például: -5, 5), az eredmény -1, 1 lesz
C. Ha $\boldsymbol{n}$ = 2 és a sorozat két eleme egymás utáni értékkel rendelkezik (például: 3, 4) az algoritmus mindig az 1, 2 értékpárt téríti vissza.
D. Az algoritmus által visszatérített két érték közül az egyik annak a leghosszabb tömbszakasznak a hossza, amely egyenlő értékeket tárol a sorozat második felében, minden $\boldsymbol{n}$ > 1 páros szám esetében.