**BABEȘ-BOLYAI UNIVERSITÄT**
**FAKULTÄT FÜR MATHEMATIK UND INFORMATIK**

# Mathe-Info-Wettbewerb - 12. April 2025

## Schriftliche Prüfung in Informatik

WICHTIGER HINWEIS:
Falls nicht anders erklärt:

- Alle arithmetischen Operationen werden mit unbegrenzten Datentypen durchgeführt (kein *Über-/Unterlauf*).
- Die Indexnummerierung aller Vektoren, Arrays und Strings beginnt bei 1.
- Alle Einschränkungen beziehen sich auf die aktuellen Parameterwerte zum Zeitpunkt des ersten Aufrufs.
- Eine Teilsequenz eines Vektors besteht aus Elementen, die aufeinanderfolgende Positionen im Vektor einnehmen.
- Wenn mehrere aufeinanderfolgende Zuweisungsanweisungen in der gleichen Zeile erscheinen, werden sie durch "; " voneinander getrennt.

**1.** Gegeben sei der Algorithmus `calcul(n, c1, c2)`, wobei $\boldsymbol{n}$ eine natürliche Zahl ist ($1 \le \boldsymbol{n} \le 10^4$) und $\boldsymbol{c1}$ und $\boldsymbol{c2}$ Ziffern sind ($0 \le \boldsymbol{c1}, \boldsymbol{c2} \le 9$).

```
Algorithm calcul(n, c1, c2):
    If n = 0 then
        Return 0
    EndIf
    If n MOD 10 = c1 then
        Return calcul(n DIV 10, c1, c2) * 10 + c2
    Else
        Return calcul(n DIV 10, c1, c2) * 10 + n MOD 10
    EndIf
EndAlgorithm
```

Was ergibt der Algorithmus für $\boldsymbol{n} = 1999$, $\boldsymbol{c1} = 1$ und $\boldsymbol{c2} = 0$?

A. 1000
B. 999
C. 1099
D. 1990

**2.** Gegeben sei der Algorithmus `ceFace(m, n)`, wobei $\boldsymbol{m}$ und $\boldsymbol{n}$ natürliche Zahlen sind ($1 \le \boldsymbol{m}, \boldsymbol{n} \le 100$):

```
 1. Algorithm ceFace(m, n):
 2.    c ← 1; i ← n
 3.    While i > 0 execute
 4.        If i MOD 2 = 1 then
 5.            c ← c * m
 6.            i ← i - 1
 7.        Else
 8.            m ← m * m
 9.            i ← i DIV 2
10.        EndIf
11.    EndWhile
12.    Return c
13. EndAlgorithm
```

Welche der folgenden Aussagen sind richtig?

A. Nach dem Aufruf `ceFace(2, 5)` liefert der Algorithmus den Wert 30.
B. Wenn der Algorithmus nach dem Aufruf `ceFace(m, n)` den Wert $\boldsymbol{x}$ liefert, gibt es kein anderes Wertepaar $\boldsymbol{m1}$, $\boldsymbol{n1}$ ($\boldsymbol{m1} \neq \boldsymbol{m}$ und $\boldsymbol{n1} \neq \boldsymbol{n}$), für das der Aufruf `ceFace(m1, n1)` den gleichen Wert $\boldsymbol{x}$ liefert.
C. Der einzige Wert von $\boldsymbol{n}$, für den die Zeile 6 nach dem Aufruf des Algorithmus `ceFace(m, n)` zweimal ausgeführt wird, ist 5.
D. Nach dem Aufruf von `ceFace(5, 8)` wird die Zeile 6 einmal ausgeführt.

**3.** Gegeben sei der Algorithmus `ceFace(b, n, a)`, wobei $\boldsymbol{b}$ und $\boldsymbol{n}$ natürliche Zahlen sind ($2 \le \boldsymbol{b}, \boldsymbol{n} \le 100$) und $\boldsymbol{a}$ ist ein Vektor mit $\boldsymbol{n}$ Elementen, natürliche Zahlen ($\boldsymbol{a}[1], \boldsymbol{a}[2], ..., \boldsymbol{a}[\boldsymbol{n}]$, $0 \le \boldsymbol{a}[\boldsymbol{i}] < \boldsymbol{b}$, for $\boldsymbol{i} = 2, 3, ..., \boldsymbol{n}$ und $0 < \boldsymbol{a}[1] < \boldsymbol{b}$):

```
Algorithm ceFace(b, n, a):
    v ← a[1]
    For i ← 2, n execute
        v ← v * b + a[i]
    EndFor
    Return v
EndAlgorithm
```

Welche der folgenden Aussagen sind richtig?

A. Der Aufruf `ceFace(2,6,[1,0,1,0,1,1])` gibt den Wert 43 zurück.
B. Der Aufruf `ceFace(9,3,[7,6,5])` liefert den Wert 626.
C. Wenn $\boldsymbol{a}[\boldsymbol{n}] = 0$ ist, gibt der Aufruf `ceFace(b, n, a)` eine gerade Zahl zurück.
D. Wenn $\boldsymbol{b1} > \boldsymbol{b2}$, dann liefert der Aufruf `ceFace(b1, n, a)` eine größere Zahl als der Aufruf `ceFace(b2, n, a)` zurück.

**4.** Gegeben sei die ganze Zahl $\boldsymbol{n}$, ($-100 \le \boldsymbol{n} \le 100$). Welcher der folgenden Ausdrücke haben den Wert *wahr* genau dann, wenn $\boldsymbol{n}$ **NICHT** zu der Menge $\{-8\} \cup \{-4, -3, ..., 8\}$ gehört?

A. `(n ≤ -8) AND (n ≥ -8) AND (n ≤ -4) AND (n ≥ 8)`
B. `(n < -8) OR ((n > -8) AND (n < -4)) OR (n > 8)`
C. `(n < -8) OR ((n > -8) OR (n < -4)) AND (n > 8)`
D. `((n < -4) AND (n ≠ -8)) OR (n > 8)`

**5.** Gegeben sei der Algorithmus `cautBin(st, dr, y, x, n)`, wobei ***st*** und ***dr*** natürliche Zahlen sind, ***x*** ein geordneter aufsteigender Vektor mit ***n*** ganzzahligen Elementen ist ($1 \le st, dr, n \le 10^4$, $x[1], x[2], ..., x[n]$, $-10^3 \le x[i] \le 10^3$ für $i = 1, 2, ..., n$) und ***y*** ist eine ganze Zahl ($-10^3 \le y \le 10^3$, $x[1] < y$), die nicht Teil des Vektors ist. Der Aufruf des Algorithmus hat die Form: `cautBin(1, n, y, x, n)`.

```
Algorithm cautBin(st, dr, y, x, n):
    If st < dr then
        mij ← (st + dr) DIV 2
        If y < x[mij] then
            Return cautBin(st, mij, y, x, n)
        Else
            Return cautBin(mij + 1, dr, y, x, n)
        EndIf
    Else
        .........
    EndIf
EndAlgorithm
```

Wodurch müssen die Auslassungspunkte ersetzt werden, damit der gegebene Algorithmus die Position der nächstgelegenen Zahl im Vektor, die größer als ***y*** ist, zurückgibt? Wenn keine solche Zahl existiert, wird -1 zurückgegeben.

A.
```
If y > x[dr] then
    Return dr + 1
Else
    Return -1
EndIf
```

B.
```
If y < x[dr] then
    Return dr
Else
    Return -1
EndIf
```

C.
```
If y > x[st] then
    Return st + 1
Else
    Return -1
EndIf
```

D.
```
If y < x[st] then
    Return st
Else
    Return -1
EndIf
```

**6.** Gegeben sei der Algorithmus `calculeaza(x, n)`, wobei ***n*** eine natürliche Zahl ist ($1 \le n \le 10^4$), und ***x*** ein Vektor mit ***n*** ganzzahligen Elementen ist ($x[1], x[2], ..., x[n]$, $-100 \le x[i] \le 100$, für $i = 1, 2, ..., n$).

```
Algorithm calculeaza(x, n):
    If n MOD 2 = 1 then
        s ← x[n]
    Else
        s ← 0
    EndIf
    For i ← 1, n - 2, 2 execute
        s ← s + x[i] + x[i + 1]
    EndFor
    Return s
EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. Nach dem Aufruf `calculeaza([3, -8, -2, 15, -1, 0, 3, 1, 3], 9)` liefert der Algorithmus den Wert 11 zurück.

B. Nach dem Aufruf `calculeaza([2, -1, 7, 5, -9, 0, 3, 1, 12], 9)`, liefert der Algorithmus den Wert 4 zurück.

C. Nach dem Aufruf `calculeaza([10, 2, 5, 78, 23, 4, 11], 7)` gibt der Algorithmus den Wert 133 zurück.

D. Nach dem Aufruf `calculeaza([-3, 8, -2, 15, -1, 10], 6)` gibt der Algorithmus den Wert 27 zurück.

**7.** Gegeben sei der Algorithmus `f(n, a, p)`, wobei ***n*** und ***p*** natürliche Zahlen sind ($1 \le n, p \le 10^5$) und ***a*** ein *n*-stelliger Vektor ist ($a[1], a[2], ..., a[n]$, $0 \le a[i] \le 9$, für $i = 1, 2, ..., n$), bei dem mindestens eine Ziffer von 0 verschieden ist:

```
Algorithm f(n, a, p):
    s ← 0
    For i ← 1, n execute
       s ← s + a[i]
    EndFor
    For i ← 1, p execute
       If s MOD 3 = 0 then
          s ← s DIV 3
       Else
          Return False
       EndIf
    EndFor
    Return True
EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. Der Algorithmus gibt *True* zurück genau dann, wenn die Summe der Elemente des Vektors ***a*** ein Vielfaches von $3^p$ ist.

B. Der Algorithmus gibt *True* zurück genau dann, wenn die Summe der Elemente des Vektors ***a*** eine Potenz von 3 ist.

C. Der Algorithmus liefert *False* nur dann, wenn die Summe der Elemente von Vektor ***a*** nicht durch 3 teilbar ist.

D. Nach dem Aufruf `f(6, [9, 1, 8, 8, 4, 6], 2)` gibt der Algorithmus *True* zurück.

**8.** Die maximale Anzahl von Kanten in einem nicht orientierten Graphen mit ***n*** Knoten und ***p*** ($0 < p \le n$) zusammenhängenden Komponenten ist:

A. $\frac{(n-p)\times(n-p+1)}{2}$

B. $(n-p)\times(n-p+1)$

C. $\frac{(n-p)\times(n-p+1)}{4}$

D. $\frac{(n-p)\times(n+p+1)}{2}$

**9.** Gegeben eine natürliche Zahl $\boldsymbol{n}$ ($10 \leq \boldsymbol{n} \leq 10^4$). Welche der folgenden Implementierungen des Algorithmus `f(n)` liefert den Spiegel der Zahl $\boldsymbol{n}$ zurück?

A.
```
Algorithm f(n):
    If n > 0 then
        Return n MOD 10 + 10 * f(n DIV 10)
    EndIf
    Return 0
EndAlgorithm
```

B.
```
Algorithm f1(n, ogl):
    If n > 0 then
        Return f1(n DIV 10, n MOD 10 + 10 * ogl)
    EndIf
    Return ogl
EndAlgorithm

Algorithm f(n):
    Return f1(n, 0)
EndAlgorithm
```

C.
```
Algorithm f(n):
    ogl ← 0
    While n > 0 execute
        ogl ← (n MOD 10) * 10 + ogl
        n ← n DIV 10
    EndWhile
    Return ogl
EndAlgorithm
```

D.
```
Algorithm f(n):
    ogl ← 0
    While n > 0 execute
        ogl ← ogl * 10 + n MOD 10
        n ← n DIV 10
    EndWhile
    Return ogl
EndAlgorithm
```

**10.** Gegeben sei der Algorithmus `ceFace(x1, y1, x2, y2, x3, y3)`, wobei (***x*1**, ***y*1**), (***x*2**, ***y*2**) und (***x*3**, ***y*3**) die Koordinaten von drei verschiedenen geometrischen Punkten sind.

```
Algorithm ceFace(x1, y1, x2, y2, x3, y3):
    t ← x1 * (y2 - y3)
    v ← x2 * (y1 - y3)
    z ← x3 * (y1 - y2)
    Return (t - v + z) ≠ 0
EndAlgorithm
```

Welche der folgenden Aussagen trifft auf den Algorithmusaufruf `ceFace(x1, y1, x2, y2, x3, y3)` zu?

A. Gibt *True* zurück, wenn die gegebenen Punkte ein echtes Dreieck bilden.
B. Gibt *False* zurück, wenn die gegebenen Punkte kollinear sind.
C. Gibt *False* zurück, wenn die gegebenen Punkte ein echtes Dreieck bilden.
D. Gibt *True* zurück, wenn die gegebenen Punkte kollinear sind.

**11.** Gegeben sei der Algorithmus `h(A, n)`, wobei $\boldsymbol{n}$ eine natürliche Zahl ist ($1 \leq \boldsymbol{n} \leq 10^3$) und $\boldsymbol{A}$ ein Vektor mit $\boldsymbol{n}$ ganzzahligen Elementen ist ($\boldsymbol{A}[1], \boldsymbol{A}[2], ..., \boldsymbol{A}[\boldsymbol{n}]$, wobei $0 \leq \boldsymbol{A}[\boldsymbol{i}] \leq 100$, für $\boldsymbol{i} = 1, 2, ..., \boldsymbol{n}$):

```
Algorithm h(A, n):
    If n = 0 then
        Return 0
    EndIf
    Return h(A, n - 1) + (A[n] MOD 2) * (A[n] MOD 10) * (n MOD 2)
EndAlgorithm
```

Nach welchen Aufrufen wird der Wert 0 zurückgegeben?

A. `h([25, 14, 35, 26, 2, 10], 6)`
B. `h([14, 25, 26, 2, 10, 35], 6)`
C. `h([12, 5, 22, 4, 32, 8, 46, 9, 54, 3], 10)`
D. `h([3, 4, 7], 3)`

**12.** Gegeben sei der Algorithmus `ceFace(n)`, wobei $\boldsymbol{n}$ eine natürliche Zahl ist ($0 \leq \boldsymbol{n} \leq 10$).

```
 1. Algorithm ceFace(n):
 2.     e ← 1
 3.     For f ← 1, n execute
 4.         s ← 0
 5.         For j ← 1, f execute
 6.             s ← s + j
 7.         EndFor
 8.         e ← e * s
 9.     EndFor
10.     Return e
11. EndAlgorithm
```

Welche der folgenden Aussagen sind richtig?

A. Nach dem Aufruf von `ceFace(5)` gibt der Algorithmus den Wert 2700 zurück.
B. Unabhängig vom Wert von $\boldsymbol{n}$ liefert der `ceFace(n)`-Algorithmus nie den Wert 0 zurück.
C. Der vom Aufruf `ceFace(9)` zurückgegebene Wert hat die gleiche Anzahl von nachgestellten Nullen wie der vom Aufruf `ceFace(10)` zurückgegebene Wert.
D. Nach dem `ceFace(10)`-Aufruf wird die sechste Zeile 45 Mal ausgeführt.

**13.** Gegeben sei der Algorithmus `ceva(n)`, wobei $n$ eine natürliche Zahl ist ($1 \leq n \leq 10^9$).

```
Algorithm aux(n):
    v1 ← 1
    v2 ← 1
    While v1 < n execute
        v3 ← v1 + v2
        v1 ← v2
        v2 ← v3
    EndWhile
    Return v1 = n
EndAlgorithm
```

```
Algorithm ceva(n):
    If aux(n) then
        Return True
    EndIf
    p ← 10
    gata ← False
    While (n DIV p ≠ 0) AND (NOT gata) execute
        nr1 ← n MOD p
        nr2 ← (n - nr1) DIV p
        If aux(nr1) then
            gata ← ceva(nr2)
        EndIf
        p ← p * 10
    EndWhile
    Return gata
EndAlgorithm
```

Wenn die ersten sechs Zahlen in der *Fibonacci-Zeichenkette* 1, 1, 2, 3, 5, 8 sind, welche der folgenden Aussagen ist wahr?

A. Der Algorithmus `ceva(n)` gibt nur dann *True* zurück, wenn $n$ eine *Fibonacci-Zahl* ist.
B. Der Algorithmus `ceva(n)` prüft, ob $n$ als Summe von *Fibonacci-Zahlen* geschrieben werden kann.
C. Der Algorithmus `ceva(n)` prüft, ob $n$ als Produkt von *Fibonacci-Zahlen* geschrieben werden kann.
D. Wenn $n = 1234589$, dann gibt der Algorithmus `ceva(n)` den Wert *True* zurück.

**14.** Gegeben sei der Algorithmus `ceFace(n, f, p)`, wobei $n$ eine natürliche Zahl ist ($0 \leq n \leq 10^{10}$), $p$ eine natürliche Zahl ist ($0 \leq p \leq 100$) und $f$ ist eine ganze Zahl ($-1 \leq f \leq 1$).

```
Algorithm ceFace(n, f, p):
    If n = 0 then
        Return f = 1
    EndIf
    c ← n MOD 10
    n ← n DIV 10
    If f = -1 then
        If c < p then
            Return ceFace(n, 0, c)
        Else
            Return False
        EndIf
    EndIf
    If f = 0 then
        If c < p then
            Return ceFace(n, 0, c)
        Else
            If c > p then
                Return ceFace(n, 1, c)
            Else
                Return False
            EndIf
        EndIf
    EndIf
    If f = 1 then
        If c > p then
            Return ceFace(n, 1, c)
        Else
            Return False
        EndIf
    EndIf
EndAlgorithm
```

Welche der folgenden Aussagen über das Ergebnis des Aufrufs `ceFace(n DIV 10, -1, n MOD 10)` sind wahr?

A. *False* wird für jeden Wert von $n < 101$ zurückgegeben.
B. Für $n = 8976532014$ wird *True* zurückgegeben.
C. Wenn $n$ mindestens zwei gleiche Ziffern enthält, wird *False* zurückgegeben.
D. Wenn $n$ nicht die Ziffer 0 enthält und der Aufruf *True* zurückgibt, dann gibt der Aufruf auch für den Spiegel von $n$ *True* zurück.

**15.** Um eine Ziffer zu bestimmen, die am häufigsten in einer Zahl vorkommt, implementieren wir drei Algorithmen: `cifreA(n)`, `cifreB(n)` und `cifreC(n)`, wobei ***n*** eine natürliche Zahl ist ($1 \leq n \leq 10^{12}$).

```
Algorithm cifreA(n):
    c ← n
    maxf ← -1; maxd ← -1
    While c > 0 execute    // (*)
        d ← c MOD 10
        copie ← n
        cnt ← 0
        While copie > 0 execute
            If copie MOD 10 = d then
                cnt ← cnt + 1
            EndIf
            copie ← copie DIV 10
        EndWhile
        If cnt > maxf then
            maxf ← cnt
            maxd ← d
        EndIf
        c ← c DIV 10
    EndWhile
    Return maxd
EndAlgorithm
```

```
Algorithm cifreB(n):
    maxf ← -1
    maxd ← -1
    For i ← 0, 9 execute    // (*)
        c ← n
        cnt ← 0
        While c > 0 execute
            If c MOD 10 = i then
                cnt ← cnt + 1
            EndIf
            c ← c DIV 10
        EndWhile
        If cnt > maxf then
            maxf ← cnt
            maxd ← i
        EndIf
    EndFor
    Return maxd
EndAlgorithm
```

```
Algorithm cifreC(n):
    maxf ← -1; maxd ← -1
    For i ← 9, 0, -1 execute
        c ← n; cnt ← 0
        While c > 0 execute
            If c MOD 10 = i then
                cnt ← cnt + 1
            EndIf
            c ← c DIV 10
        EndWhile
        If cnt > maxf then
            maxf ← cnt
            maxd ← i
        EndIf
    EndFor
    Return maxd
EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. `cifreA(123453) = cifreB(123453) = cifreC(123453)`

B. `cifreA(123456) = cifreB(123456) = cifreC(123456)`

C. Es gibt mindestens eine Zahl ***n***, für die die drei Algorithmen drei verschiedene Werte zurückliefern.

D. Für jede Zahl ***n*** wird der mit (*) gekennzeichnete **`While`**-Zyklus im Algorithmus `cifreA(n)` weniger oft durchgelaufen als der mit (*) gekennzeichnete **`For`**-Zyklus im Algorithmus `cifreB(n)`.

**16.** Gegeben sei der Algorithmus `getSomeMax(n, x)`, ***n*** ist eine natürliche Zahl ($1 \leq n \leq 10^3$) und ***x*** ein Vektor mit ***n*** ganzzahligen Elementen ($x[1], x[2], ..., x[n]$, wobei $-10^3 \leq x[i] \leq 10^3$, für $i = 1, 2, ..., n$). Der Algorithmus `zero(k)` gibt einen Vektor mit ***k*** Elementen zurück, die alle gleich Null sind.

```
Algorithm getSomeMax(n, x):
    y ← zero(n + 1)
    For i ← 1, n execute
        y[i + 1] ← y[i] + x[i]
    EndFor
    sm ← y[2]
    For i ← 2, n execute
        For j ← i, n execute
            s ← y[j] - y[i - 1]
            If s > sm then
                sm ← s
            EndIf
        EndFor
    EndFor
    Return sm
EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. Wenn ***n*** = 1 ist, ist der vom Algorithmus `getSomeMax(n, x)` zurückgegebene Wert der Wert von ***x***[1].

B. Der Wert, den der Algorithmus im Falle des Aufrufs `getSomeMax(8, [5, 7, -4, 6, -3, -2, 6, -7])` zurückgibt, ist 10.

C. Ist ***n*** = 100 und ***x*** = `[1, 2, 3, ..., 99, 100]`, so ist der vom Algorithmus `getSomeMax(n, x)` zurückgegebene Wert gleich mit 4950.

D. Wenn alle Werte im Vektor ***x*** streng negativ sind, gibt der Algorithmus `getSomeMax(n, x)` das größte Element im Vektor zurück.

**17.** Gegeben sei der Algorithmus `afla(n, x)`, wobei ***n*** eine natürliche Zahl ist ($3 \le n \le 10^4$) und ***x*** ein Vektor mit ***n*** ganzzahligen Elementen ist (***x***[1], ***x***[2], ..., ***x***[***n***], -100 ≤ ***x***[***i***] ≤ 100, für ***i*** = 1, 2, ..., ***n***):

```
1. Algorithm afla(n, x):
2.     M1 ← x[1]; M2 ← x[2]; M3 ← x[3]
3.     For i ← 1, n execute
4.         If x[i] > M1 then
5.             M3 ← M2
6.             M2 ← M1
7.             M1 ← x[i]
8.         Else
9.             If x[i] > M2 then
10.                M3 ← M2
11.                M2 ← x[i]
12.            Else
13.                If x[i] > M3 then
14.                    M3 ← x[i]
15.                EndIf
16.            EndIf
17.        EndIf
18.    EndFor
19.    Return M1, M2, M3
20.EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. Nach dem Aufruf `afla(6, [1, 2, 3, 4, 5, 6])` gibt der Algorithmus 6, 5, 4 zurück.

B. Wenn die Anweisungen in den Zeilen 8 und 12 durch **`EndIf`** ersetzt und die Anweisungen in den Zeilen 16 und 17 gelöscht würden, würde der Algorithmus dasselbe Ergebnis liefern wie der ursprüngliche Algorithmus.

C. Wenn ***M*1**, ***M*2** und ***M*3** zu Beginn die Werte ***x***[3], ***x***[2] bzw. ***x***[1] annehmen würden, würde der Algorithmus das gleiche Ergebnis wie der ursprüngliche Algorithmus liefern.

D. Wenn in Zeile 3 an der Stelle von `For i ← 1, n execute` `For i ← 4, n execute` stehen würde, würde der Algorithmus dasselbe Ergebnis liefern wie der ursprüngliche Algorithmus.

**18.** Gegeben sei der Algorithmus `ceFace(A, n)`, wobei ***n*** eine natürliche Zahl ist ($1 \le n \le 20$) und ***A*** eine quadratische Matrix mit ***n*** Zeilen und ***n*** Spalten ist, deren Einträge natürliche Zahlen sind (***A***[1][1], ***A***[1][2], ..., ***A***[***n***][***n***], wobei 0 ≤ ***A***[***i***][***j***] ≤ 200, für ***i*** = 1, 2, ..., ***n*** und ***j*** = 1, 2, ..., ***n***).

```
Algorithm ceFace(A, n):
    // Început partea 1
    For i ← 1, n execute
        For j ← i + 1, n execute
            temp ← A[i][j]
            A[i][j] ← A[j][i]
            A[j][i] ← temp
        EndFor
    EndFor
    // Sfârșit partea 1

    // Început partea 2
    For i ← 1, n execute
        For j ← 1, n DIV 2 execute
            temp ← A[i][j]
            A[i][j] ← A[i][n - j + 1]
            A[i][n - j + 1] ← temp
        EndFor
    EndFor
    // Sfârșit partea 2
EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. Nach Ausführung des Algorithmus `ceFace(A, 3)` wird die Matrix ***A*** = $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$, zu $\begin{matrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{matrix}$ geändert.

B. Wenn die Eingabematrix ***A*** die Einheitsmatrix der Ordnung 3 ist, dann ändert sie sich nicht durch die Ausführung des Algorithmus `ceFace(A, 3)`.

C. Der Algorithmus `ceFace(A, n)` wendet eine 90°-Drehung nach rechts auf die gegebene Matrix an und verändert sie entsprechend.

D. Wenn man den Teil des Algorithmus zwischen `Început partea 1` und `Sfârșit partea 1` mit dem Teil zwischen `Început partea 2` und `Sfârșit partea 2` vertauscht, liefert der Algorithmus `ceFace(A, n)` das gleiche Ergebnis wie der ursprüngliche Algorithmus.

**19.** Gegeben sei der Algorithmus `ceFace(n)`, wobei ***n*** eine natürliche Zahl ist ($0 \le n \le 200$).

```
1. Algorithm ceFace(n):
2.     e ← 0
3.     For i ← 1, n execute
4.         If i MOD 2 = 0 then
5.             e ← e - 2 * i * i
6.         Else
7.             e ← e + 2 * i * i
8.         EndIf
9.     EndFor
10.    Return e
11. EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. Für jede gerade Zahl ***n*** gibt der Algorithmus einen negativen Wert zurück.

B. Der Algorithmus berechnet den Wert des Ausdrucks $0 + 1 * 2 - 2 * 4 + 3 * 6 - 4 * 8 + ... + (-1)^{n-1} * n * 2 * n$

C. Wenn der Algorithmus `ceFace(n)` einen negativen Wert liefert, ist ***n*** eine gerade Zahl.

D. Es gibt nur einen Wert für ***n***, für den die Anweisung in Zeile 7 genau 7 Mal ausgeführt wird.

**20.** Gegeben sei der Algorithmus `ceFace(n, x)`, wobei ***n*** eine natürliche Zahl ist ($2 \leq n \leq 10^3$) und ***x*** ein Vektor mit ***n*** ganzzahligen Elementen ist (***x***[1], ***x***[2], ..., ***x***[***n***], -100 ≤ ***x***[***i***] ≤ 100, für ***i*** = 1, 2, ..., ***n***). Der Algorithmus `zero(k)` gibt einen Vektor mit ***k*** Elementen zurück, die alle gleich Null sind. Die Algorithmen `minim(n, x)`, `maxim(n, x)` geben den minimalen bzw. maximalen Wert des Vektors ***x*** mit ***n*** Elementen zurück

```
01. Algorithm ceFace(n, x):
02.     min ← minim(n, x)
03.     max ← maxim(n, x)
04.     r ← max – min + 1
05.     y ← zero(r)
06.     For i ← 1, n execute
07.         y[x[i] – min + 1] ← y[x[i] – min + 1] + 1
08.     EndFor
09.     idx ← 1
10.     For i ← 1, r execute
11.         While y[i] > 0 execute
12.             x[idx] ← i + min – 1
13.             idx ← idx + 1
14.             y[i] ← y[i] - 1
15.         EndWhile
16.     EndFor
17. EndAlgorithm
```

Welche der folgenden Aussagen sind richtig?

A. Wenn der Vektor ***x*** auch negative Zahlen enthält, wird der Algorithmus versuchen, auf nicht existierende Positionen im Vektor ***y*** zuzugreifen.

B. Wenn wir die Anweisungen in den Zeilen 9 und 10 durch die folgende Anweisungsfolge ersetzen, würde der Algorithmus `ceFace(n, x)` das gleiche Ergebnis liefern wie der ursprüngliche Algorithmus.

```
x[1] ← min
idx ← 2
y[1] ← y[1] – 1
For i ← 2, r execute
```

C. Nach dem Aufruf `ceFace(2, [5, 8])` wird der Vektor ***x*** zu: ***x*** = `[6, 9]`.

D. Nach der Ausführung des Algorithmus `ceFace(n, x)` werden die Elemente des Vektors ***x*** eine Permutation der Elemente des initialen Vektors darstellen.

**21.** Gegeben sei der Algorithmus `p(x, n, a, b, c, d)`, wobei ***x*** ein Vektor mit ***n*** ($0 \leq n \leq 100$) ganzzahligen Elementen ist (***x***[1], ***x***[2], ..., ***x***[***n***], wobei -100 ≤ ***x***[***i***] ≤ 100, für ***i*** = 1, 2, ..., ***n***), und ***a***, ***b***, ***c***, und ***d*** ganze Zahlen sind (0 ≤ ***a***, ***b***, ***c***, ***d*** ≤ 100).

```
Algorithm p(x, n, a, b, c, d):
    If n = 0 then
        Return a = b AND c = d
    EndIf
    p1 ← p(x, n - 1, a + x[n], b, c * x[n], d)
    p2 ← p(x, n - 1, a,  b + x[n], c, d * x[n])
    Return p1 OR p2
EndAlgorithm
```

Wenn man weiß, dass ***x*** = `[2, 9, 5, 6, 8, 4, 1, 2, 5, 3, 4, 1, 9, 6, 8, 3]` ist, welche der folgenden Aussagen ist wahr?

A. Nach dem Aufruf `p(x, 16, 0, 0, 1, 1)` gibt der Algorithmus *True* zurück.

B. Nach dem Aufruf `p(x, 16, 0, 0, 1, 1)` gibt der Algorithmus *False* zurück.

C. Mit dem Aufruf `p(x, 16, 0, 0, 1, 1)` tritt der Algorithmus in einen unendlichen Zyklus ein.

D. Nach dem Aufruf `p(x, 16, 0, 0, 1, 1)` liefert der Algorithmus für jede Permutation der Elemente des Vektors ***x*** das gleiche Ergebnis.

**22.** Gegeben seien die Algorithmen `rec(n, x, i, j)` und `ceFace(n, x)`, wobei ***n*** eine natürliche Zahl ist ($1 \leq n \leq 10^3$) und ***x*** ein Vektor mit ***n*** ganzzahligen Elementen (***x***[1], ***x***[2], ..., ***x***[***n***], -100 ≤ ***x***[***k***] ≤ 100, für ***k*** = 1, 2, ..., ***n***), und ***i*** und ***j*** sind ganze Zahlen im Intervall [0, ***n***]. Der Algorithmus `maxim(a, b)` gibt das größere von ***a*** und ***b*** zurück.

```
Algorithm rec(n, x, i, j):
    If i = n then
        Return 0
    EndIf
    a ← rec(n, x, i + 1, j)
    b ← 0
    If j = 0 then
        b ← 1 + rec(n, x, i + 1, i)
    Else
        If x[i] > x[j] then
            b ← 1 + rec(n, x, i + 1, i)
        EndIf
    EndIf
    Return maxim(a, b)
EndAlgorithm
```

```
Algorithm ceFace(n, x):
    Return rec(n, x, 1, 0)
EndAlgorithm
```

Welche der folgenden Aussagen sind wahr?

A. Für einen streng aufsteigenden geordneten Vektor ***x*** gibt der Algorithmus `ceFace(n, x)` den Wert ***n*** zurück**.**

B. Die Zeitkomplexität des Algorithmus beträgt im ungünstigsten Fall $O(n^2)$.

C. Nach dem Aufruf `ceFace(8, [10, 15, 9, 30, 21, 50, 42, 60])` liefert der Algorithmus den Wert 5.

D. Nach dem Aufruf von `ceFace(2, [3, 2])` liefert der Algorithmus den Wert 1.

**23.** Eine Zahl ***n*** wird als *speziell* bezeichnet, wenn sie nur die Zahlen 2, 3 und 5 als Primteiler hat. Spezielle Zahlen sind zum Beispiel 1 ($1 = 2^0 * 3^0 * 5^0$), 12 ($12 = 2^2 * 3$) oder 30 ($30 = 2 * 3 * 5$). Der „`zero(k)`-Algorithmus gibt einen Vektor mit ***k*** Elementen gleich 0 zurück. Welche der Befehlsfolgen in den Antworten A, B, C, D sollten in den Algorithmus `special(n)` anstelle der Auslassungspunkte eingefügt werden, damit der Algorithmus die ***n-te*** spezielle Zahl liefert, wobei ***n*** eine natürliche Zahl ist ($1 \le n \le 10^5$)?

```
Algorithm special(n):
    v ← zero(n)
    v[1] ← 1; c2 ← 1; c3 ← 1; c5 ← 1
    nr ← 1
    While nr < n execute
        val1 ← v[c2] * 2
        val2 ← v[c3] * 3
        val3 ← v[c5] * 5
        If val1 ≤ val2 AND val1 ≤ val3 then
            elem ← val1
            c2 ← c2 + 1
        Else
            If val2 ≤ val1 AND val2 ≤ val3 then
                elem ← val2; c3 ← c3 + 1
            Else
                elem ← val3
                c5 ← c5 + 1
            EndIf
        EndIf
        ..........
    EndWhile
    Return v[n]
EndAlgorithm
```

A.
```
v[nr] ← elem
nr ← nr + 1
```

B.
```
If v[nr] < elem then
    v[nr + 1] ← elem
    nr ← nr + 1
EndIf
```

C.
```
nr ← nr + 1
v[nr] ← elem
```

D.
```
tmp ← nr
While elem < v[tmp] AND tmp ≥ 1 execute
    v[tmp + 1] ← v[tmp]
    tmp ← tmp - 1
EndWhile
v[tmp + 1] ← elem
nr ← nr + 1
```

**24.** Bei einer natürlichen Zahl ***n*** ($0 \le n \le 2^{31}$) soll die Anzahl der Bits mit dem Wert ***k*** $\in \{0, 1\}$ in der Basis-2-Darstellung der Zahl ***n*** bestimmt werden, die mit genau 32 Bits dargestellt wird. In den Algorithmen werden die Bitoperationen `&` (**`UND`**), `<<` (Linksverschiebung der Darstellung) und `>>` (Rechtsverschiebung der Darstellung) mit den folgenden Bedeutungen verwendet:

- Wenn ***x*** und ***y*** zwei natürliche Zahlen sind, dann wendet ***x* & *y*** die bitweise **`UND`** Operation auf ihrer Binärdarstellung an: Jedes Bit im Ergebnis ist nur dann 1, wenn beide entsprechenden Bits in ***x*** und ***y*** gleich 1 sind; andernfalls ist es 0.
- Wenn ***x*** eine natürliche Zahl ist, ist die Operation ***x* << *i*** gleichbedeutend mit der Multiplikation von ***x*** mit 2 genau ***i*** mal, und die Operation ***x* >> *i*** ist gleichbedeutend mit der ganzzahligen Division von ***x*** durch 2 genau ***i*** mal.

Welcher der folgenden Algorithmen liefert den gewünschten Wert?

A.
```
Algorithm countBits_A(n, k):
    count ← 0
    For i ← 0, 31 execute
        If ((n & (1 << i)) >> i) = k then
            count ← count + 1
        EndIf
    EndFor
    Return count
EndAlgorithm
```

B.
```
Algorithm countBits_B(n, k):
    count ← 0
    While n > 0 execute
        If (n & 1) = 1 then
            count ← count + 1
        EndIf
        n ← n >> 1
    EndWhile
    If k = 0 then
        count ← 32 – count
    EndIf
    Return count
EndAlgorithm
```

C.
```
Algorithm countBits_C(n, k):
    If n = 0 then
        If k = 0 then
            Return 32
        Else
            Return 0
        EndIf
    Else
        If (n & 1) = k then
            Return 1 + countBits_C(n >> 1, k)
        Else
            Return countBits_C(n >> 1, k)
        EndIf
    EndIf
EndAlgorithm
```

D.
```
Algorithm countBits(n, k, poz):
    If poz < 0 then
        Return 0
    Else
        If ((n & (1 << poz)) >> poz) = k then
            Return 1 + countBits(n, k, poz - 1)
        Else
            Return countBits(n, k, poz - 1)
        EndIf
    EndIf
EndAlgorithm

Algorithm countBits_D(n, k):
    Return countBits(n, k, 31)
EndAlgorithm
```

BABEŞ-BOLYAI UNIVERSITÄT

FAKULTÄT FÜR MATHEMATIK UND INFORMATIK

Mathe-Info-Wettbewerb - 12. April 2025

Schriftliche Prüfung in Informatik

PUNKTEANZAHL & LÖSUNGEN

**ANFANGSPUNKTEANZAHL**: 10 punkte

| | | |
|---|---|---|
| **1** | B | 3.75 punkte |
| **2** | D | 3.75 punkte |
| **3** | ABD | 3.75 punkte |
| **4** | BD | 3.75 punkte |
| **5** | BD | 3.75 punkte |
| **6** | C | 3.75 punkte |
| **7** | AD | 3.75 punkte |
| **8** | A | 3.75 punkte |
| **9** | BD | 3.75 punkte |
| **10** | AB | 3.75 punkte |
| **11** | BC | 3.75 punkte |
| **12** | AB | 3.75 punkte |
| **13** | D | 3.75 punkte |
| **14** | AD | 3.75 punkte |
| **15** | AC | 3.75 punkte |
| **16** | AC | 3.75 punkte |
| **17** | A | 3.75 punkte |
| **18** | AC | 3.75 punkte |
| **19** | BC | 3.75 punkte |
| **20** | D | 3.75 punkte |
| **21** | AD | 3.75 punkte |
| **22** | D | 3.75 punkte |
| **23** | B | 3.75 punkte |
| **24** | ABD | 3.75 punkte |