BABEŞ-BOLYAI UNIVERSITÄT
FAKULTÄT FÜR MATHEMATIK UND INFORMATIK

Wettbewerb Mathe-Info - März 2021
Schriftliche Prüfung in Informatik

WICHTIG ZU BEACHTEN:
Wenn keine anderen Spezifikationen vorliegen, wird davon ausgegangen, dass alle arithmetischen Operationen für unbegrenzte Datentypen ausgeführt werden (es gibt keinen Überlauf / Unterlauf).
Man nimmt an, dass die Indizierung aller Arrays/Sequenzen bei 1 beginnt.

1. Gegeben sei der folgende Ausdruck, wobei *a* eine natürliche Zahl ist.

```
((a < 4) ODER (a < 5)) UND (a > 2)
```

Für welche Werte von *a* ist dieser Ausdruck ***WAHR***?

A. a = 3
B. a = 4
C. a = 2
D. Der Ausdruck ist niemals wahr

2. Gegeben sei der unten definierten Unteralgorithmus, der als Parameter eine Folge **v** bestehend aus **n** natürliche, von Null verschiedene Zahlen (v[1], v[2], …, v[n]) und die ganze Zahl n ($1 \leq n \leq 10000$) bekommt.

```
Subalgorithm f(v, n):
    x ← 0
    For i ← 1, n do
        c ← v[i]
        While c MOD 3 = 0 do
            x ← x + 1
            c ← c DIV 3
        EndWhile
    EndFor
    return x
EndSubalgorithm
```

Geben Sie an welche der folgenden Aussagen wahr sind:

A. Der Unteralgorithmus gibt die Anzahl der durch 3 teilbaren Zahlen der Folge v zurück
B. Der Unteralgorithmus gibt die größte Zahl *k* zurück, so dass v[1] * v[2] * … * v[n] durch 3^k teilbar ist
C. Der Unteralgorithmus gibt die größte Zahl *k* zurück, so dass v[1] + v[2] + … + v[n] durch 3^k teilbar ist
D. Der Unteralgorithmus gibt die Summe der durch 3 teilbaren Zahlen der Folge v zurück

3. Gegeben sei der folgende Ausdruck, wobei *x* eine positive, natürliche Zahl ist.

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

Welche der folgenden Aussagen sind wahr?

A. Der Ausdruck hat den Wert 1 für jede positive natürliche Zahl x.
B. Der Ausdruck hat den Wert 1 genau dann, wenn x gerade ist.
C. Der Ausdruck hat den Wert 1 genau dann, wenn x ungerade ist.
D. Es existiert eine natürliche Zahl x für die der Ausdruck einen Wert, der echt größer als 1 ist hat.

4. Gegeben sei der unten definierten Unteralgorithmus **`verarbeiten`**, der als Parameter eine Folge **x** bestehend aus **n** reelle, von Null verschiedene Zahlen (x[1], x[2], …, x[n]) und die ganze Zahl n ($1 \leq n \leq 10000$) bekommt.
Der Operator **`/`** beschreibt die reelle Division (ex. **`3/2=1,5)`**.

```
Subalgorithm verarbeiten(x, n):
    p ← 1
    For k ← 1, n - 1 do
        p ← p + 1
        For i ← 1, n - 1 do
            If x[i] > x[i + 1] then
                x[i] ← x[i] * x[i + 1]
                x[i + 1] ← x[i] / x[i + 1]
                x[i] ← x[i] / x[i + 1]
            EndIf
        EndFor
    EndFor
    n ← p
EndSubalgorithm
```

Welche der folgenden Aussagen beschreiben die Veränderung der Folge *x* nach dem Aufruf des Unteralgorithmus `verarbeiten(x, n)`?

A. Die Elemente der Folge x bleiben unverändert
B. Die Elemente der Folge x werden monoton fallend geordnet
C. Die Elemente der Folge x werden monoton wachsend geordnet
D. Die Zahl n wird mit einer Einheit dekrementiert

5. Gegeben sei der Unteralgorithmus `berechne(a, n)`, der als Parameter eine Folge **a** bestehend aus **n** natürliche Zahlen (a[1], a[2], …, a[n]) und die ganze Zahl n ($1 \leq n \leq 10000$) bekommt.

```
Subalgorithm berechne(a, n):
    If n = 0 then
        return 0
    else
        return a[n] * (a[n] MOD 2) + berechne(a, n - 1)
    EndIf
EndSubalgorithm
```

Für welche Werte von **n** und der Folge **a** wird die Funktion `berechne(a, n)` den Wert 10 zurückgeben?

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. Gegeben sei der Unteralgorithmus `berechne(v, n)`, der als Parameter eine Folge **v** bestehend aus **n** natürliche Zahlen (v[1], v[2], …, v[n]) und die ganze Zahl n ($1 \leq n \leq 10000$) bekommt.

```
Subalgorithm berechne(v, n):
    m ← 0
    x ← 0
    s ← 0
    For i ← 1, n do
        s ← s + v[i]
        m ← m + (s MOD 2 + x) MOD 2
        x ← s MOD 2
    EndFor
    return m
EndSubalgorithm
```

Geben Sie an welche der folgenden Ausdrücke wahr sind:

A. Der Unteralgorithmus berechnet und gibt die Summe der ungeraden Zahlen der Folge v zurück
B. Der Unteralgorithmus berechnet und gibt die Summe der geraden Zahlen der Folge v zurück
C. Der Unteralgorithmus berechnet und gibt die Anzahl der ungeraden Zahlen der Folge v zurück
D. Der Unteralgorithmus berechnet und gibt die Anzahl der geraden Zahlen der Folge v zurück

7. Gegeben sei der Unteralgorithmus `magic(x)`, wobei **x** eine natürliche Zahl ($1 \leq x \leq 32000$) ist.

```
Subalgorithm magic(x):
    st ← 1
    dr ← x
    While st ≤ dr do
        mj ← (st + dr) DIV 2
        If mj * mj = x then
            return true
        EndIf
        If mj * mj < x then
            st ← mj + 1
        else
            dr ← mj - 1
        EndIf
    EndWhile
    return false
EndSubalgorithm
```

Geben Sie an welche der folgenden Ausdrücke wahr sind:

A. Der Unteralgorithmus überprüft ob eine Quadratzahl kleiner als x existiert.
B. Der Unteralgorithmus zählt die Primfaktoren von x.
C. Der Unteralgorithmus überprüft ob x eine Primzahl ist.
D. Der Unteralgorithmus überprüft ob x eine Quadratzahl ist.

8. Gegeben sei der Unteralgorithmus `wasMacht(n)`, wobei n eine natürliche Zahl ($1 \leq n \leq 10000$) ist.

```
Subalgorithm wasMacht(n):
    a ← n
    b ← 0
    While a ≠ 0 do
      b ← b * 10 + a MOD 10
      a ← a DIV 10
    EndWhile
    If n = b then
      return true
    else
      return false
    EndIf
EndSubalgorithm
```

Geben Sie an welche der folgenden Ausdrücke wahr sind:

A. Der Unteralgorithmus überprüft ob n eine Primzahl ist.
B. Der Unteralgorithmus überprüft ob x ein Palindrom ist.
C. Der Unteralgorithmus gibt als Antwort immer wahr zurück.
D. Der Unteralgorithmus überprüft ob x durch 10 teilbar ist.

9. Gegeben sei der Unteralgorithmus `berechne(a,b)`, wobei **a** und **b** natürliche Zahlen ($1 \leq a, b \leq 10000$) sind.

```
Subalgorithm berechne(a, b):
    x ← 1
    For i ← 1, b do
        x ← (x MOD 10) * a
    EndFor
    return x
EndSubalgorithm
```

Geben Sie an welche der folgenden Ausdrücke wahr sind:

A. Für a = 2021 und b = 2021, gibt der Unteralgorithmus den Wert 2021 zurück.
B. Für alle Aufrufe des Unteralgorithmus mit a = 2021 und $1 \leq b \leq 10000$, ist der zurückgegebene Wert gleich mit 2021.
C. Für a = 7777 und b = 2021, gibt der Unteralgorithmus den Wert 7777 zurück.
D. Für alle Aufrufe des Unteralgorithmus mit $1 \leq a \leq 10000$ und b = 2021, ist der zurückgegebene Wert gleich dem Wert von a.

10. Wieviele Elemente befinden sich auf den zwei Diagonalen einer quadratischen Matrix mit **n** Zeilen und **n** Spalten ($10 \leq n \leq 1000$)? Man zählt nur die Elemente, die sich auf jeweils verschiedene Positionen befinden.

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

11. Welche der folgenden logischen Ausdrücke sind WAHR für $\boldsymbol{a} = 1$ und $\boldsymbol{b} = 0$?

A. `NOT (((a > 0) UND (b < 1)) ODER (a > 1))`
B. `((b > 0) UND (b < 1)) ODER ((a > 0) UND (a < 2))`
C. `(NOT (a > b)) ODER (NOT (b > 0))`
D. `(a > 0) ODER ((b > 0) UND (b < 0)) ODER (a < 1)`

12. Die Unteralgorithmen $\texttt{berechne}_i\texttt{(e, n)}$, $1 \leq \mathbf{i} \leq 4$, bekommen als Parameter eine Matrix **e** mit **n** Zeilen und **n** Spalten (e[1][1], …, e[1][n], e[2][1], …, e[n][n]) und eine natürliche Zahl **n** ($1 \leq \mathbf{n} \leq 1000$). Wählen Sie diejenigen Antworten, die die Definition des Unteralgorithmus $\texttt{berechne}_i\texttt{(e, n)}$ beinhalten, der ein anderes Ergebnis als alle andere drei Varianten hat, das heißt so dass $\texttt{berechne}_i\texttt{(e, n)} \neq \texttt{berechne}_j\texttt{(e, n)}$ $\forall$ **e**, **n**, **j**, $1 \leq \mathbf{j} \leq 4$, $i \neq j$ (**e** und **n** entsprechen der gegebenen Spezifikation).

A.

```
Subalgorithm berechne1(e, n):
    s ← 0
    For i ← 1, n do
        s ← s + e[i][i]
    EndFor
    return s
EndSubalgorithm
```

B.

```
Subalgorithm berechne2(e, n):
    s ← 0
    For i ← 1, n do
        For j ← 1, n, do
            If i = j then
                s ← s + e[i][j]
            EndIf
        EndFor
    EndFor
    return s
EndSubalgorithm
```

C.

```
Subalgorithm berechne3(e, n):
    s ← 0
    i ← 1
    While i ≤ n do
        s ← s + e[i][i]
        i ← i + 1
    EndWhile
    return s
EndSubalgorithm
```

D.

```
Subalgorithm berechne4(e, n):
    s ← 0
    For i ← 1, n do
        For j ← i + 1, n do
            If i = j then
                s ← s + e[i][j]
            EndIf
        EndFor
    EndFor
    return s
EndSubalgorithm
```

13. Gegeben sei der Unteralgorithmus `wasMacht(a,b)`, wobei **a** und **b** natürliche Zahlen ($1 \leq \mathbf{a} < \mathbf{b} \leq 10000$) sind.

```
Subalgorithm wasMacht(a, b):
    m ← a
    While b MOD m > 0 do
        m ← m + 1
    EndWhile
    return m
EndSubalgorithm
```

Was wird der Aufruf `wasMacht(47, 100)` zurückgeben?

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

14. Gegeben sei der Unteralgorithmus `angabe(n)`, wobei **n** eine natürliche Zahl ($0 \leq n \leq 10000$) ist.

```
Subalgorithm angabe(n):
    Write n
    If n > 0 then
        angabe(n - 1)
        Write n
    EndIf
EndSubalgorithm
```

Was wird der Aufruf `angabe(4)` zurückgeben?

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

15. Welche der folgenden Zahlenbasen ***x*** erfüllen die Bedingung $232_{(x)} \leq 67_{(10)}$?

A. x = 5
B. x = 3
C. x = 4
D. x = 6

16. Der Unteralgorithmus `bewegeNull(a, n)` erhält als Parameter eine Folge **a** ganzer Zahlen, (a[1], a[2], …, a[n]) und die ganze Zahl **n** ($1 \leq n \leq 10000$). Der Unteralgorithmus bewegt die Nullwerte an das Ende der Folge, die relative Ordnung der von Null verschiedenen Elemente behaltend. Zum Beispiel, falls a die Folge `[4, 0, 2, 5, 1, 0, 7, 11, 0, 3]` ist, sind die Elemente von a nach dem Aufruf des Unteralgorithmus `[4, 2, 5, 1, 7, 11, 3, 0, 0, 0]`. Welche der folgenden Implementationen des Unteralgorithmus `bewegeNull(a, n)` sind korrekt??

A.

```
Subalgorithm bewegeNull(a, n):
    s ← TRUE
    While s = TRUE do
        s ← FALSE
        For i ← 1, n - 1 do
            If a[i] = 0 then
               tmp ← a[i]
               a[i] ← a[i + 1]
               a[i + 1] ← tmp
               s ← TRUE
            EndIf
        EndFor
    EndWhile
EndSubalgorithm
```

B.

```
Subalgorithm bewegeNull(a, n):
    c ← 0
    For i ← 0, n do
        If a[i] = 0 then
            c ← c + 1
        EndIf
    EndFor
    i ← n
    While c > 0 do
        a[i] ← 0
        i ← i - 1
        c ← c - 1
    EndWhile
EndSubalgorithm
```

C.

```
Subalgorithm bewegeNull(a, n):
    d ← 0
    i ← 1
    While i + d ≤ n do
        While (i + d ≤ n) AND (a[i + d] = 0) do
            d ← d + 1
        EndWhile
        If i + d ≤ n then
            a[i] ← a[i + d]
            i ← i + 1
        EndIf
    EndWhile
    While i ≤ n do
        a[i] ← 0
        i ← i + 1
    EndWhile
EndSubalgorithm
```

D.

```
Subalgorithm bewegeNull(a, n):
    i ← 1
    f ← n
    While i < f do
        While (i < f) AND (a[i] ≠ 0) do
            i ← i + 1
        EndWhile
        While (i < f) AND (a[f] = 0) do
            f ← f - 1
        EndWhile
        If i < f then
            tmp ← a[i]
            a[i] ← a[f]
            a[f] ← tmp
        EndIf
    EndWhile
EndSubalgorithm
```

17. Gegeben sei die Folge ***X***=1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7…, in der jede Zahl ***n*** auf nacheinanderfolgende Stellen ***n*** mal erscheint. Angenommen, dass das erste Element der Folge auf der Stelle 1 vorkommt, auf welchen Stellen wird 21 vorkommen?

A. Auf Stellen aus dem Intervall [210,230]
B. Auf Stellen aus dem Intervall [211,231]
C. Auf Stellen aus dem Intervall [212,232]
D. Auf Stellen aus dem Intervall [209,229]

18. Gegeben sei der Unteralgorithmus `f(a, b)`, wobei ***a*** und ***b*** ganze Zahlen (-10000 ≤ **a**, **b** ≤ 10000) sind.

```
Subalgorithm f(a, b):
    Write  FMI”
    If (a = 0) OR (b = 0) then
        return 1
    EndIf
    If a > b then
        return f(a - b * b, a * (a - b) - b * (a - b))
    EndIf
    If a ≤ b then
        return f(b - a * a, a * (a - b) - b * (a - b))
    EndIf
EndSubalgorithm
```

Geben Sie an wie oft der Text *FMI* geschrieben wird, nach der Ausführung von:

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

A. 8 mal
B. 6 mal
C. 3 mal
D. unendlich oft

19. Gegeben sei der rekursive Unteralgorithmus `wasMacht(n, i)`, wobei ***n*** eine natürliche Zahl ($2 \leq \boldsymbol{n} \leq 1000$) ist.

```
Subalgorithm wasMacht(n, i):
    If i * i > n then
        return 0
    EndIf
    If i * i = n then
        return i
    EndIf
    If n MOD i = 0 then
        return i + n DIV i + wasMacht(n, i + 1)
    else
        return wasMacht(n, i + 1)
    EndIf
EndSubalgorithm
```

Geben Sie an welche der folgenden Aussagen nach dem Aufruf von `wasMacht(n, 2)` wahr sind:

A. Der Unteralgorithmus berechnet und gibt den zweifachen Wert der Summe der echten Teiler der Zahl `n` zurück.
B. Der Unteralgorithmus berechnet und gibt die Summe der echten Teiler der Zahl `n` zurück.
C. Der Unteralgorithmus berechnet und gibt die Summe der echten und unechten Teiler der Zahl `n` zurück.
D. Der Unteralgorithmus überprüft ob `n` eine Quadratzahl ist. In diesem Fall berechnet und gibt die quadratische Wurzel dieser Zahl zurück. Ansonsten wird 0 zurückgegeben.

20. Gegeben sei der Unteralgorithmus `wasMacht(T, n, e)`, der als Parameter eine Folge **T** aus **n** monoton wachsend geordnete natürliche Zahlen (T[1], T[2], …, T[n]) und die natürliche Zahlen **n** und **e** ($1 \leq \mathbf{n}, \mathbf{e} \leq 10000$) bekommt.

```
Subalgorithm wasMacht(T, n, e):
    If e MOD 2 = 0 then
        a ← 1
        b ← n
        While a ≤ b do
            m ← (a + b) DIV 2
            If e < T[m] then
                b ← m - 1
            else
                If e > T[m] then
                    a ← m + 1
                else
                    return true
                EndIf
            EndIf
        EndWhile
        return false
    else
        c ← 1
        While c ≤ n do
            If e = T[c] then
                return true
            EndIf
            c ← c + 1
        EndWhile
        return false
    EndIf
EndSubalgorithm
```

Geben Sie an welche der folgenden Aussagen wahr sind:

A. Der Unteralgorithmus überprüft nicht, dass die Zahl **e** sich auf einer geraden Stelle in der Folge **T** befindet.
B. Der Unteralgorithmus überprüft ob die Zahl **e** sich in der Folge **T** befindet. Falls die Zahl **e** ungerade ist, wird der Suchalgorithmus die binäre Suche sein.
C. Der Unteralgorithmus überprüft ob die Zahl **e** sich in der Folge **T** befindet. Falls die Zahl **e** gerade ist, wird der Suchalgorithmus die binäre Suche sein.
D. Der Unteralgorithmus überprüft ob die Zahl **e** sich in der Folge **T** befindet nur wenn die Zahl **e** ungerade ist.

21. Wir wollen gleichseitige Dreiecke nur mit * (Stern) und . (Punkt) darstellen. Das folgende Beispiel zeigt ein Dreieck dessen Seite n=5 Sterne lang ist. Dafür sind 12 Sterne und 23 de Punkte notwendig.

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

Welche der folgenden Aussagen sind wahr?

A. Für n=2 sind genau 3 Sterne und 4 Punkte notwendig.
B. Für n=7 sind genau 18 Sterne und 52 Punkte notwendig.
C. Für n=7 sind genau 18 Sterne und 48 Punkte notwendig.
D. Für n=15 sind genau 42 Sterne und 288 Punkte notwendig.

22. Eine Folge bestehend aus **n** Zeichen ist ein Antipalindrom falls alle Zeichenpaare mit gleicher Entfernung zum ersten und letzten Zeichen der Folge jeweils verschieden sind (mit Ausnahme des Mittelzeichens falls **n** ungerade ist). Zum Beispiel sind *asdfg und xlxe* Antipalindrome, aber *asdsg* ist keins.

Gegeben sei der Unteralgorithmus `antipalindrom(s, links, rechts)` der als Parameter eine Folge **s** mit **n** ($1 \leq \mathbf{n} \leq 10000$) Zeichen (s[1], s[2], …, s[n]), und die natürlichen Zahlen `links` und `rechts` bekommt.

Welche der folgenden Implementationen wird genau dann *wahr* für den Aufruf `antipalindrom(s, 1, n)` zurückgeben, wenn die Folge **s** ein Antipalindrom ist?

A.

```
Subalgorithm antipalindrom(s, links, rechts):
    If links = rechts then
        return true
    else
        erster ← s[links]
        letzter ← s[rechts]
        If erster = letzter then
            return false
        else
            return antipalindrom(s, links + 1, rechts - 1)
        EndIf
    EndIf
EndSubalgorithm
```

B.

```
Subalgorithm antipalindrom(s, links, rechts):
    If links ≥ rechts then
        return true
    EndIf
    erster ← s[links]
    letzter ← s[rechts]
    If erster = letzter then
        return false
    else
        return antipalindrom(s, links + 1, rechts - 1)
    EndIf
EndSubalgorithm
```

C.

```
Subalgorithm antipalindrom(s, links, rechts):
    If links > rechts then
        return true
    else
        erster ← s[links]
        letzter ← s[dreapta]
        If erster ≠ letzter then
            return false
        else
            return antipalindrom(s, links + 1, rechts - 1)
        EndIf
    EndIf
EndSubalgorithm
```

D.

```
Subalgorithm antipalindrom(s, links, rechts):
    If links > rechts then
        return true
    EndIf
    erster ← s[links]
    letzter ← s[rechts]
    If erster ≠ letzter then
        return true
    EndIf
    return antipalindrom(s, links + 1, rechts - 1)
EndSubalgorithm
```

23. Gegeben sei der Unteralgorithmus `ordo(n, a)` der als Parameter ein natürliche Zahl **n** ($1 \leq \mathbf{n} \leq 10000$) und eine Folge **a** mit 2n Elemente natürliche Zahlen (a[1], a[2], …, a[2n]) bekommt.

Angenommen, die Anzahl der geraden Elemente der Folge **a** ist gleich mit der Anzahl der ungeraden Elemente, welcher der folgenden Unteralgorithmen ordnet die Elemente der Folge **a** so dass die ungeraden Elemente ungerade Indizes und die geraden Elemente gerade Indizes haben?

A.

```
Subalgorithm ordo(n, a):
  For i ← 1, 2 * n - 1 do
    If a[i] MOD 2 ≠ i MOD 2 then
      For j ← i + 1, 2 * n do
        If a[j] MOD 2 ≠ j MOD 2 then
          a[i] ← a[i] + a[j]
          a[j] ← a[i] - a[j]
          a[i] ← a[i] - a[j]
        EndIf
      EndFor
    EndIf
  EndFor
EndSubalgorithm
```

B.

```
Subalgorithm ordo(n, a):
  For i ← 1, 2 * n - 1 do
    If a[i] MOD 2 ≠ i MOD 2 then
      For j ← i + 1, 2 * n do
        If (a[i] MOD 2 ≠ i MOD 2) AND
           (a[j] MOD 2 ≠ j MOD 2) then
          a[i] ← a[i] + a[j]
          a[j] ← a[i] - a[j]
          a[i] ← a[j] - a[i]
        EndIf
      EndFor
    EndIf
  EndFor
EndSubalgorithm
```

C.

```
Subalgorithm ordo(n, a):
  For i ← 1, 2 * n - 1 do
    If a[i] MOD 2 ≠ i MOD 2 then
      For j ← i + 1, 2 * n do
        If (a[i] MOD 2 ≠ i MOD 2) AND
           (a[j] MOD 2 ≠ j MOD 2) AND
           (a[i] MOD 2 ≠ a[j] MOD 2) then
          a[i] ← a[i] + a[j]
          a[j] ← a[i] - a[j]
          a[i] ← a[i] - a[j]
        EndIf
      EndFor
    EndIf
  EndFor
EndSubalgorithm
```

D.

```
Subalgorithm ordo(n, a):
  For i ← 1, 2 * n - 1 do
    For j ← i + 1, 2 * n do
      If (a[j] MOD 2 = 0) AND
         ((a[j] MOD 2 ≠ 0) OR (a[j] MOD 2 ≠ 0)) AND
         (a[j] MOD 2 = 0) then
        a[i] ← a[i] + a[j]
        a[j] ← a[i] - a[j]
        a[i] ← a[i] - a[j]
      EndIf
    EndFor
  EndFor
EndSubalgorithm
```

24. Wir wollen eine Folge mit **n** ($1 \leq n \leq 1000$) Werte in **k** ($1 \leq k \leq n$) benachbarte Unterfolgen gleicher Länge partitionieren (jedes Folgenelement gehört zu genau einer Unterfolge). Falls **n** durch **k** nicht teilbar ist kann die Differenz der Längen beliebiger Unterfolgen höchstens 1 sein.
Gegeben seien vier Varianten um die Indizes des ersten Elementes der Unterfolge **j** ($1 \leq j \leq k$) zu generieren. Falls die Folgenelemente ausgehend von 1 nummeriert sind, welche der folgenden Varianten erfüllen die obige Bedingung?

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. Sei $b_n b_{n-1}...b_0$ die binäre Darstellung der natürlichen Zahl *B*, $2021 \leq B \leq 2021^{2021}$. Welche der folgenden Aussagen sind wahr?

A. Falls der Wert von $b_0-b_1+b_2-b_3+.....+(-1)^n * b_n$ Null ist, dann ist B durch 3 teilbar
B. Falls der Wert von $b_0-b_1+b_2-b_3+....+(-1)^n * b_n$ durch 3 teilbar ist, dann ist B durch 3 teilbar
C. B ist durch 3 teilbar falls die Summe der binären Ziffern durch 3 aber nicht durch 9 teilbar ist
D. Falls B durch 3 teilbar ist, dann ist $b_0-b_1+b_2-b_3+....+(-1)^n * b_n$ durch 3 teilbar

26. Gegeben sei der Unteralgorithmus `prefix(n)`, welcher für eine gegebene natürliche Zahl **n** ($9 < n < 10^{20}-1$), das längste Präfix zurückgibt, der sich auch innerhalb der Zahl selbst befindet (mit Ausnahme der ersten und der letzten Ziffer). Der Unteralgorithmus gibt die Länge dieses Präfixes zurück.

Beispiel:

für ***n*** = 12133121, ist das Präfix 12 und hat Länge 2.
für ***n*** = 34534536, ist das Präfix 3453 und hat Länge 4.
für ***n*** = 1223, gibt es kein solches Präfix (wir betrachten die Länge als 0).

Falls die Indizierung der Folgen mit 1 beginnt, welche der folgenden Varianten sind korrekte Implementationen des Unteralgorithmus `prefix(n)`?

A.

```
Subalgorithm prefix(n):
  nr ← n
  c ← 0
  p ← 1
  While nr > 0 do
    c ← c + 1
    nr ← nr DIV 10
    p ← p * 10
  EndWhile

  f1 ← 100
  f2 ← p DIV 100
  k ← 1
  ok ← 0
  While ok = 0 do
    n1 ← n DIV f1
    f3 ← 10
    For i ← 1 ,k do
      n2 ← (n DIV f3) MOD f2
      If n1 = n2 then
        ok ← 1
        return c - k - 1
      EndIf
      f3 ← f3 * 10
    EndFor
    f1 ← f1 * 10
    f2 ← f2 DIV 10
    k ← k + 1
  EndWhile
  return -1
EndSubalgorithm
```

B.

```
Subalgorithm prefix(n):
  c ← [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
  nr ← n
  p ← 0
  While nr > 0 do
    c[p + 1] ← nr MOD 10
    nr ← nr DIV 10
    p ← p + 1
  EndWhile
  For i ← 1, p - 2 do
    For j ← p – 1, i + 1, -1 do
      ok ← 1
      For k ← 0, i - 1 do
        If c[p - 1 - k] ≠ c[j - k] then
          ok ← 0
        EndIf
      EndFor
      If ok = 1 then
        return i
      EndIf
    EndFor
  EndFor
  return -1
EndSubalgorithm
```

C.

```
Subalgorithm prefix(n):
  c ← [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
  nr ← n
  p ← 0
  While nr > 0 do
    c[p + 1] ← nr MOD 10
    nr ← nr DIV 10
    p ← p + 1
  EndWhile
  For i ← p - 2, 1, -1 do
    For j ← p – 1, i + 1, -1 do
      ok ← 1
      For k ← 0, i - 1 do
        If c[p - k] ≠ c[j - k] then
          ok ← 0
        EndIf
      EndFor
      If ok = 1 then
        return j
      EndIf
    EndFor
  EndFor
  return 0
EndSubalgorithm
```

D.

```
Subalgorithm prefix(n):
  nr ← n
  c ← 0
  p ← 1
  While nr > 0 do
    c ← c + 1
    nr ← nr DIV 10
    p ← p * 10
  EndWhile
  f1 ← p DIV 10
  f2 ← 10
  k ← c - 2
  ok ← -1
  For t ← 1, c - 2 do
    n1 ← n DIV f1
    f3 ← 10
    For i ← 1, k do
        n2 ← (n DIV f3) MOD f2
        If n1 = n2 then
            If ok < c - k - 1 then
                ok ← c - k - 1
            EndIf
        EndIf
        f3 ← f3 * 10
    EndFor
    f1 ← f1 DIV 10
    f2 ← f2 * 10
    k ← k - 1
  EndFor
  If ok < 0 then:
    return 0
  EndIf
  return ok
EndSubalgorithm
```

27. Gegeben sei die folgende Tabelle mit 16 Einträge (4 Zeilen bezeichnet mit 1, 2, 3, 4 und 4 Spalten bezeichnet mit A, B, C, D). Manche Einträge sind konstante Werte (z.B. Eintrag B3), andere, die mit "=" anfangen enthalten das Ergebnis eines arithmetischen Ausdrucks mit 2 Glieder. Jedes Glied ist entweder konstant oder, falls das Glied mit dem Symbol $ anfängt, eine Referenz zum Wert eines anderen Eintrags. Zum Beispiel, Eintrag A4 enthält das Ergebnis der arithmetischen Subtraktion Operation: aus dem konstanten Wert 5 wird der Wert aus dem Eintrag A2 subtrahiert. Für einen Eintrag i, bezeichnen wir mit X(i) die minimale Anzahl der Einträge (inklusive die, die konstante Werte enthalten) deren Werte bekannt sein müssen vor der Ausrechnung des Wertes aus dem Eintrag i. Ähnlich, bezeichnen wir mit Y(i) die maximale Anzahl der Einträge (inklusive diejenigen, die konstante Werte enthalten, mit Ausnahme des Eintrags i) deren Werte ohne dem Wert des Eintrags i berechnet werden können. Welche der folgenden Aussagen sind für die Werte von X(A2) und Y(A2) wahr?

	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 und Y(A2) = 1
B. X(A2) = 5 und Y(A2) = 13
C. X(A2) = 6 und Y(A2) = 4
D. X(A2) = X(C4) und Y(A2) = Y(B2) + 1

28. Der Unteralgorithmus `vereinfache(nr, num)` berechnet den unkürzbaren Bruch **aux1 / aux2** mit der Eigenschaft **aux1 / aux2 = nr / num** (**aux1**, **aux2**, **nr**, **num** natürliche Zahlen, **num, aux2** $\neq 0$).

```
Subalgorithm vereinfache(nr, num):
  d ← funktion(nr, num)
  aux1 ← nr DIV d
  aux2 ← num DIV d
EndSubalgorithm
```

Welche der folgenden Varianten des Unteralgorithmus `funktion(a, b)` sind richtig?

A.

```
Subalgorithm funktion(a, b):
 d ← 1
 While true do
   If (a MOD d = 0) AND (b MOD d = 0) then
     return d
   EndIf
   d ← d + 1
 EndWhile
EndSubalgorithm
```

B.

```
Subalgorithm funktion(a, b):
 While b ≠ 0 do
   c ← a MOD b
   a ← b
   b ← c
 EndWhile
 return a
EndSubalgorithm
```

C.

```
Subalgorithm funktion(a, b):
 While a ≠ b do
   If a > b then
     a ← a - b
   else
     b ← b - a
   EndIf
 EndWhile
 return a
EndSubalgorithm
```

D.

```
Subalgorithm funktion(a, b):
  d ← a
  While ((a MOD d ≠ 0) OR (b MOD d ≠ 0)) do
    d ← d - 1
  EndWhile
  return d
EndSubalgorithm
```

29. Gegeben sei der rekursive Unteralgorithmus `fibonacci(n)`, wobei **n** eine natürliche Zahl ($1 \leq n \leq 100$) ist. Man bestimme wie oft die Nachricht "`hier`" für den Aufruf `fibonacci(n)` (wir zählen den initialen Aufruf und alle rekursiven Aufrufe mit) dargestellt wird.

```
Subalgorithm fibonacci(n):
  If n ≤ 1 then
    return 1
  else
    Write  hier"
    return fibonacci(n - 1) + fibonacci(n - 2)
  EndIf
EndSubalgorithm
```

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

30. Gegeben sei der Ausdruck: $E(x) = a_0 + a_1*x + a_2*x^2 + a_3*x^3 + a_5*x^5$, wobei a_0, a_1, a_2, a_3, a_5 und x reelle, von Null verschiedene Zahlen sind. Die minimale Anzahl der notwendigen Multiplikationen, um den Wert des Ausdruckes ***E(x)*** zu berechnen ist:

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