UNIVERSITÄT BABEŞ-BOLYAI
MATHEMATIK UND INFORMATIK FAKULTÄT

Aufnahmeprüfung und Mathe-Info Wettbewerb - Modell

Schriftliche Prüfung in Informatik

1. Das Unterprogramm `generieren(n)` bearbeitet eine natürliche Zahl ***n*** ($0 < n < 100$).

```
Subalgorithm generieren(n):
   nr ← 0
   For i ← 1, 1801 execute
      benutzt_i ← false
   EndFor
   While not benutzt_n execute
      summe ← 0, benutzt_n ← true
      While (n ≠ 0) execute
         ziffer ← n MOD 10, n ← n DIV 10
         summe ← summe + ziffer * ziffer * ziffer
      EndWhile
      n ← summe, nr ← nr + 1
   EndWhile
   return nr
EndSubalgorithm
```

Welche ist die Auswirkung des Unterprogramms?

A. es berechnet wiederholt die Summe der Kubikzahlen der Ziffern der Zahl ***n*** (Summe der „Ziffern hoch 3") bis die Summe gleich mit der Zahl ***n*** ist, und es gibt die Anzahl der ausgeführten Wiederholungen zurück.
B. es berechnet die Summe der Kubikzahlen der Ziffern der Zahl ***n*** und gibt diese Summe zurück.
C. es berechnet die Summe der Kubikzahlen der Ziffern der Zahl ***n***, ersetzt die Zahl ***n*** mit der erhaltenen Summe und gibt diese Summe zurück.
D. es berechnet die Anzahl der Ersetzungen von ***n*** mit der Summe der Kubikzahlen seiner Ziffern bis man einen Wert, der sich wiederholt, oder die Zahl selber erhält, und gibt diesen Wert zurück.

2. Sei ***s*** eine Sequenz von ***k*** Elementen vom Typ boolesch und das Unterprogramm `auswertung(s, k, i)`, wobei ***k*** und ***i*** natürliche Zahlen sind ($0 \leq i \leq k \leq 100$).

```
Subalgorithm auswertung(s, k, i)
   If i ≤ k then
      If s_i then
         return s_i
      else
         return (s_i or auswertung(s, k, i + 1))
      EndIf
   else
      return false
   EndIf
EndSubalgorithm
```

Bestimme wie oft das Unterprogramm `auswertung(s, k, i)` in der folgenden Anweisungssequenz sich selbst aufruft:

```
s ← (false, false, false, false, false, false, true, false, false, false)
k ← 10
i ← 3
auswertung(s, k, i)
```

A. 3 Mal.
B. genau so oft wie in der folgenden Anweisungssequenz:

```
s ← (false, false, false, false, false, false, false, true)
k ← 8
i ← 4
auswertung(s, k, i)
```

C. 6 Mal.
D. unendlich oft.

3. Gegeben sei ein Unterprogramm `ausdruck(n)`, wobei ***n*** eine natürliche Zahl ist ($1 \leq \boldsymbol{n} \leq 10000$).

```
Subalgorithm ausdruck(n):
   If n > 0 then
      If n MOD 2 = 0 then
         return -n * (n + 1) + ausdruck(n - 1)
      else
         return n * (n + 1) + ausdruck(n - 1)
      EndIf
   else
      return 0
   EndIf
EndSubalgorithm
```

Bestimme die mathematische Form des Ausdruckes *E(n)*, der von diesem Unterprogramm berechnet wird:

A. $E(n) = 1* 2 - 2 * 3 + 3 * 4 + ... + (-1)^{n+1} * n * (n + 1)$
B. $E(n) = 1* 2 - 2 * 3 + 3 * 4 + ... + (-1)^{n} * n * (n + 1)$
C. $E(n) = 1* 2 + 2 * 3 + 3 * 4 + ... + (-1)^{n+1} * n * (n + 1)$
D. $E(n) = 1* 2 - 2 * 3 - 3 * 4 - ... - (-1)^{n} * n * (n + 1)$

4. Ein Integer Datentyp auf ***x*** Bits repräsentiert, wobei ***x*** eine natürliche streng positive Zahl ist, kann als Werte ganze Zahlen aus dem folgenden Intervall speichern:

A. $[0, 2^x]$
B. $[0, 2^{x-1}-1]$
C. $[-2^{x-1},2^{x-1}-1]$
D. $[-2^x,2^x-1]$

5. Gegeben sei das Unterprogramm `f(a, b)`:

```
Subalgorithm f(a, b):
       If a > 1 then
              return b * f(a - 1, b)
       Else
              return b * f(a + 1, b)
       EndIf
EndSubalgorithm
```

Gebe an, wie oft die Funktion `f` in der Ausführung des folgenden Anweisungsblocks sich selbst aufruft:

```
a ← 4
b ← 3
c ← f(a, b)
```

A. 4 Mal.
B. 3 Mal.
C. Unendlich oft.
D. keinmal.

6. Gegeben sei der folgende logische Ausdruck: **(NOT *Y* OR *Z*) OR (*X* AND *Y*)**. Wähle die Werte für ***X***, ***Y***, ***Z*,** sodass das Auswertungsergebnis des Ausdruckes *wahr* ist:

A. ***X*** ← *falsch*; ***Y*** ← *falsch*; ***Z*** ← *falsch*;
B. ***X*** ← *falsch*; ***Y*** ← *wahr*; ***Z*** ← *falsch*;
C. ***X*** ← *wahr*; ***Y*** ← *falsch*; ***Z*** ← *wahr*;
D. ***X*** ← *falsch*; ***Y*** ← *wahr*; ***Z*** ← *wahr*;

7. Geben Sie an welche der folgenden Ausdrücke genau dann wahr ist wenn die natürliche Zahl ***n*** durch 3 teilbar ist und auf 4 oder 6 endet:

A. `n DIV 3 = 0 and (n MOD 10 = 4 or n MOD 10 = 6)`
B. `n MOD 3 = 0 and (n MOD 10 = 4 or n MOD 10 = 6)`
C. `(n MOD 3 = 0 and n MOD 10 = 4) or (n MOD 3 = 0 and n MOD 10 = 6)`
D. `(n MOD 3 = 0 and n MOD 10 = 4) or n MOD 10 = 6`

8. Gegeben sei der folgende Subalgorithmus:

```
Subalgorithm f(a):
    If a ≠ 0 then
        return a + f(a - 1)
    else
        return 0
    EndIf
EndSubalgorithm
```

Welche der folgenden Aussagen sind falsch?

A. falls a negativ ist, dann gibt der Subalgorithmus den Wert 0 zurück.
B. der Wert, der von f zurückgegeben wird ist a * (a + 1) / 4.
C. der Subalgorithmus berechnet die Summe der natürlichen Zahlen kleiner oder gleich a.
D. der Aufruf f(-5) läuft unendlich.

9. Gegeben sei das folgende Unterprogramm:

```
Subalgorithm SA9(a):
    If a < 50 then
        If a MOD 3 = 0 then
            return SA9(2 * a - 3)
        else
            return SA9(2 * a - 1)
        EndIf
    else
        return a
    EndIf
EndSubalgorithm
```

Für welche Werte des Eingabeparameters ***a*** wird das Unterprogramm den Wert 61 zurückgeben?

A. 16
B. 61
C. 4
D. 31

10. Gegeben sei das Unterprogramm `verarbeitung(v, k)`, wobei ***v*** eine Sequenz mit ***k*** natürlichen Zahlen ist ($1 \leq \boldsymbol{k} \leq 1\,000$).

```
Subalgorithm verarbeitung(v, k)
   i ← 1, n ← 0
   While i ≤ k and v_i ≠ 0 execute
      y ← v_i, c ← 0
      While y > 0 execute
         If y MOD 10 > c then
            c = y MOD 10
         EndId
         y ← y DIV 10
      EndWhile
      n ← n * 10 + c
      i ← i + 1
   EndWhile
   return n
EndSubalgorithm
```

Bestimme für welche Werte von ***v*** und **k** das Unterprogramm den Wert 928 zurückgibt.

A. ***v*** = (194, 121, 782, 0) und ***k*** = 4
B. ***v*** = (928) und ***k*** = 1
C. ***v*** = (9, 2, 8, 0) und ***k*** = 4
D. ***v*** = (8, 2, 9) und ***k*** = 3

11. Gegeben sei der folgende logische Ausdruck `(X OR Z) AND (NOT X OR Y)`. Wählen Sie entsprechende Werte für ***X***, ***Y***, ***Z*** so dass die Evaluierung des Ausdrucks `TRUE` liefert:

A. ***X*** ← `FALSE`; ***Y*** ← `FALSE`; ***Z*** ← `TRUE`;
B. ***X*** ← `TRUE`; ***Y*** ← `FALSE`; ***Z*** ← `FALSE`;
C. ***X*** ← `FALSE`; ***Y*** ← `TRUE`; ***Z*** ← `FALSE`;
D. ***X*** ← `TRUE`; ***Y*** ← `TRUE`; ***Z*** ← `TRUE`;

12. Gegeben sei folgendes Programm:

C Version

```
#include <stdio.h>

int prelVector(int v[], int *n) {
   int s = 0; int i = 2;
   while (i <= *n) {
      s = s + v[i] - v[i - 1];
      if (v[i] == v[i - 1])
         *n = *n - 1;
      i++;
   }
   return s;
}


int main(){
  int v[8];
  v[1] = 1; v[2] = 4; v[3] = 2;
  v[4] = 3; v[5] = 3; v[6] = 10;
  v[7] = 12;
  int n = 7;
  int ergebnis = prelVector(v, &n);
  printf("%d;%d", n, ergebnis);
  return 0;
}
```

C++ Version

```
#include <iostream>
using namespace std;
int prelVector(int v[], int&n) {
   int s = 0; int i = 2;
   while (i <= n) {
      s = s + v[i] - v[i - 1];
      if (v[i] == v[i - 1])
         n--;
      i++;
   }
   return s;
}


int main(){
   int v[8];
   v[1] = 1; v[2] = 4; v[3] = 2;
   v[4] = 3; v[5] = 3; v[6] = 10;
   v[7] = 12;
   int n = 7;
   int  ergebnis = prelVector(v, n);
   cout << n <<";" <<  ergebnis;
   return 0;
}
```

Pascal Version

```
type vector=array [1..10] of integer;
function prelVector(v: vector;
              var n: integer): integer;
   var s, i: integer;
   begin
      s := 0; i := 2;
      while (i <= n) do
        begin
          s := s + v[i] - v[i - 1];
          if (v[i] = v[i - 1]) then
              n := n - 1;
          i := i + 1;
       end;
      prelVector := s;
   end;
var n,  ergebnis:integer; v:vector;
begin
   n := 7;
   v[1] := 1; v[2] := 4; v[3] := 2;
   v[4] := 3; v[5] := 3; v[6] := 10;
   v[7] := 12;
    ergebnis:= prelVector(v,n);
   write(n, ';',  ergebnis);
end.
```

Geben Sie an welches Ergebnis nach der Durchführung des Programms angezeigt wird.

A. 7;11
B. 6;9
C. 7;9
D. 7;12

13. Gegeben sei der folgende in Pseudocode geschriebener Algorithmus:

```
read a
For i=1, a-1 do
       For j=i+2, a do
              If i+j>a-1 then
                     write a, ‘ ‘, i, ‘ ‘, j
                     start new line
              EndIf
       EndFor
EndFor
```

Wieviele Lösungspaare werden nach der Durchführung des Algorithmus mit a=8 angezeigt?

A. 13
B. 15
C. 20
D. Keine der anderen Antworten ist korrekt.

14. Welches der folgenden Unterprogramme gibt das größte Vielfach der natürlichen Zahl ***a*** an, welches kleiner oder gleich mit der natürlichen Zahl ***b*** ist ($0 < \boldsymbol{a} < 10\,000$, $0 < \boldsymbol{b} < 10\,000$, $\boldsymbol{a} < \boldsymbol{b}$)?

A.
```
Subalgorithm f(a, b):
  c ← b
  While c MOD a = 0 execute
    c ← c – 1
  EndWhile
  return c
EndSubalgorithm
```

B.
```
Subalgorithm f(a, b):
  If a < b then
    return f(2 * a, b)
  else
    If a = b then
      return a
    else
      return b
    EndIf
  EndIf
EndSubalgorithm
```

C.
```
Subalgorithm f(a, b):
  return (b DIV a) * a
EndSubalgorithm
```

D.
```
Subalgorithm f(a, b):
  If b MOD a = 0 then
    return b
  EndIf
  return f(a, b - 1)
EndSubalgorithm
```

15. Seien alle Zeichenfolgen der Länge $\boldsymbol{l} \in \{1, 2, 3\}$ bestehend aus den Buchstaben aus der Menge {*a*, *b*, *c*, *d*, *e*}. Wie viele dieser Zeichenfolgen haben die Elemente in streng absteigender Reihenfolge geordnet und haben zusätzlich eine ungerade Anzahl von Vokalen. (*a* und *e* sind Vokale)

A. 14
B. 7
C. 81
D. 78

16. Das Unterprogramm `gehört(x, a, n)` überprüft, ob eine natürliche Zahl ***x*** zu der Menge ***a*** mit ***n*** Elementen gehört; ***a*** ist eine Sequenz mit ***n*** Elementen und stellt eine Menge von natürlichen Zahlen dar ($1 \leq \boldsymbol{n} \leq 200$, $1 \leq \boldsymbol{x} \leq 1000$). Seien die weiter unten beschriebene Unterprogramme `vereinigung(a, n, b, m, c, p)` und `berechnung(a, n, b, m, c, p)`, wobei ***a***, ***b*** und ***c*** Sequenzen sind, die Mengen von natürlichen Zahlen mit ***n***, ***m*** und beziehungweise ***p*** Elementen darstellen ($1 \leq \boldsymbol{n} \leq 200$, $1 \leq \boldsymbol{m} \leq 200$, $1 \leq \boldsymbol{p} \leq 400$). Die Eingabeparameter sind ***a***, ***n***, ***b, m*** und ***p***, und die Ausgabeparameter sind ***c*** und ***p***.

```
1.      Subalgorithm vereinigung(a, n, b, m, c, p):
2.        If n = 0 then
3.          For i ← 1, m execute
4.            p ← p + 1
5.            c_p ← b_i
6.          EndFor
7.        else
8.          If not gehört(a_n, b, m) then
9.            p ← p + 1
10.           c_p ← a_n
11.         EndIf
12.         vereinigung(a, n - 1, b, m, c, p)
13.       EndIf
14.     EndSubalgorithm
```

```
1.  Subalgorithm berechnung(a, n, b, m, c,
    p):
2.     p ← 0
3.     vereinigung(a, n, b, m, c, p)
4.  EndSubalgorithm
```

Bestimme welche der folgenden Aussagen immer wahr sind:

A. wenn die Menge ***a*** ein einziges Element enthält, dann entsteht bei dem Aufruf des Unterprogramms `berechnung(a, n, b, m, c, p)` ein unendlicher Zyklus.
B. wenn die Menge ***a*** 4 Elemente enthält, dann wird bei dem Aufruf des Unterprogramms `berechnung(a, n, b, m, c, p)` die Anweisung von der Linie 12 des Unterprogramms `vereinigung` viermal ausgeführt.
C. wenn die Menge ***a*** 5 Elemente enthält, dann wird bei dem Aufruf des Unterprogramms `berechnung(a, n, b, m, c, p)` die Anweisung von der Linie 2 des Unterprogramms `vereinigung` fünfmal ausgeführt.
D. wenn die Menge ***a*** die gleichen Elemente wie die Menge ***b*** enthält, dann wird die Menge ***c*** nach der Ausführung des Unterprogramms `berechnung(a, n, b, m, c, p)` dieselbe Anzahl von Elementen wie die Menge ***a*** haben.

17. Gegeben sei das Unterprogramm `berechnung(n)`, wobei ***n*** eine natürliche Zahl ist ($1 \leq n \leq 10000$).

```
Subalgorithm berechnung(n):
  x ← 0, z ← 1
  While z ≤ n execute
    x ← x + 1
    z ← z + 2 * x
    z ← z + 1
  EndWhile
  return x
EndSubalgorithm
```

Welche der folgenden Aussagen sind **falsch**?

A. Falls $n < 8$, dann gibt `berechnung(n)` den Wert 3 zurück.
B. Falls $n \geq 85$ und $n < 100$, dann gibt `berechnung(n)` den Wert 9 zurück.
C. Das Unterprogramm berechnet und gibt die Anzahl der Quadratzahlen zurück, die streng positiv und streng kleiner als ***n*** sind.
D. Das Unterprogramm berechnet und gibt den ganzen Teil der Wurzel der Zahl ***n*** zurück.

18. Sei ***mat*** eine quadratische $n \times n$ - Matrix (n – natürliche ungerade Zahl, $3 \leq n \leq 100$) und das Unterprogramm `setzeB(mat, n, i, j)`, das die Elemente bestimmter Positionen aus der Matrix ***mat*** auf das Zeichen 'b' setzt. Die Parameter i und j sind natürliche Zahlen ($1 \leq i \leq n$, $1 \leq j \leq n$).

```
Subalgorithm setzeB(mat, n, i, j):
   If i ≤ n DIV 2 then
      If j ≤ n - i then
         mat[i][j] ← 'b'
         setzeB(mat, n, i, j + 1)
      else
         setzeB(mat, n, i + 1, i + 2)
      EndIf
   EndIf
EndSubalgorithm
```

Bestimme wie oft das Unterprogramm `setzeB(mat, n, i, j)` sich selbst aufruft, wenn man folgende Anweisungssequenz hat:

```
n ← 7, i ← 2, j ← 4
setzeB(mat, n, i, j)
```

A. 5 Mal
B. Genau so oft wie in der Anweisungssequenz
```
n ← 9, i ← 3, j ← 5
setzeB(mat, n, i, j)
```
C. 10 Mal
D. unendlich oft

19. Gegeben sei der Unteralgorithmus `berechnung(a, b)` mit den Eingangsparameter a und b natürliche Zahlen, $1 \leq a \leq 1000$, $1 \leq b \leq 1000$.

```
1.      Subalgorithm berechnung(a, b):
2.         If a ≠ 0 then
3.            return berechnung(a DIV 2, b + b) + b * (a MOD 2)
4.         EndIf
5.         return 0
6.      EndSubalgorithm
```

Welche der unteren Aussagen sind falsch?

A. falls a und b gleich sind, der Unteralgorithmus gibt den Wert von a zurück.
B. falls $a = 1000$ und $b = 2$, ruft sich der Unteralgorithmus 10 mal selbst auf.
C. Der vom Unteralgorithmus berechneter Wert ist $a / 2 + 2 * b$
D. Der Befehl der 5. Zeile wird nur einmal durchgeführt.

20. Gegeben sei der Unteralgorithmus `PrimFaktoren(n, d, k, x)` welches die k Primfaktoren einer natürlichen Zahl n berechnet, die Suche der Primfaktoren beginnt beim Wert d. Die Eingangsparameter sind die natürlichen Zahlen n, d und k, die Ausgangsparameter sind die Folge x der k Primfaktoren ($1 \leq n \leq 10000$, $2 \leq d \leq 10000$, $0 \leq k \leq 10000$).

```
Subalgorithm PrimFaktoren(n, d, k, x):
   If n MOD d = 0 then
      k ← k + 1
      x[k] ← d
   EndIf
   While n MOD d = 0 do
      n ← n DIV d
   EndWhile
   If n > 1 then
      PrimFaktoren(n, d + 1, k, x)
   EndIf
EndSubalgorithm
```

Bestimmen Sie wie oft der Unteralgorithmus `PrimFaktoren(n, d, k, x)` sich selbst in der folgenden Anweisungsequenz aufruft:

```
n ← 120
d ← 2
k ← 0
PrimFaktoren(n, d, k, x)
```

A. 3 Mal.
B. 5 Mal.
C. 6 Mal.
D. gleich oft wie in folgender Anweisungsequenz:

```
n ← 750
d ← 2
k ← 0
PrimFaktoren(n, d, k, x)
```

21. Gegeben die natürlichen Zahlen ***m*** und ***n*** ($0 \leq \boldsymbol{m} \leq 10$, $0 \leq \boldsymbol{n} \leq 10$) und der Unteralgorithmus `Ack(m, n)`, welches den Wert der Ackermann Funktion für die Werte ***m*** und ***n*** berechnet.

```
Subalgorithm Ack(m, n)
  If m = 0 then
      return n + 1
  else
    If m > 0 and n = 0 then
      return Ack(m - 1, 1)
    else
      return Ack(m - 1, Ack(m, n - 1))
    EndIf
  EndIf
EndSubalgorithm
```

Geben Sie an wie oft der Unteralgorithmus `Ack(m, n)` sich selbst aufruft während der Durchführung der folgenden Anweisungsequenz:

```
m ← 1, n ← 2
Ack(m, n)
```

A. 7 Mal.
B. 5 Mal.
C. 10 Mal.
E. gleich oft wie in folgender Anweisungsequenz:

```
m ← 1, n ← 3
Ack(m, n)
```

22. Wir definieren die *Abschneidungsoperation* einer natürlichen Zahl mit ***k*** Ziffern $\overline{c_1 c_2 \dots c_k}$ folgendermaßen: $Abschneiden(\overline{c_1 c_2 \dots c_k}) = \begin{cases} 0, falls\ k < 2; \\ \overline{c_1 c_2}, ansonsten \end{cases}$.

Geben Sie an welche der folgenden Unteralgorithmen die *Summe der Abschneidungen* der Elemente einer Folge ***x*** bestehend aus ***n*** natürliche Zahlen kleiner als 1 000 000 (***n*** – natürliche Zahl, $1 \leq n \leq 1\,000$)? Zum Beispiel, falls ***n*** = 4 und ***x*** = (213, 7, 78 347, 22), dann ist die Summe der Abschneidungen 21 + 0 + 78 + 22 = 121.

A.

```
Subalgorithm summeAbschneidungen(n, x)
s ← 0
While n > 0 do
    If x[n] > 9 then
      While x[n] > 99 do
         x[n] ← x[n] DIV 10
      EndWhile
      s ← s + x[n]
    EndIf
    n ← n - 1
 EndWhile
 return s
EndSubalgorithm
```

B.

```
Subalgorithm summeAbschneidungen(n, x)
 s ← n
 While n > 0 do
    If x[n] > 9 then
       While x[n] > 99 do
         x[n] ← x[n] DIV 10
       EndWhile
       s ← s + x[n]
    EndIf
    n ← n - 1
 EndWhile
 return s
EndSubalgorithm
```

C.

```
Subalgorithm summeAbschneidungen(n, x)
    s ← 0
    While n > 0 do
       If x[n] > 9 then
         While x[n] > 99 do
              x[n] ← x[n] DIV 10
              s ← s + x[n]
         EndWhile
      EndIf
      n ← n - 1
    EndWhile
    return s
EndSubalgorithm
```

D.

```
Subalgorithm summeAbschneidungen(n, x)
    s ← 0
    While x[n] > 99 do
       x[n] ← x[n] DIV 10
    EndWhile
       s ← s + x[n]
    return s
 EndSubalgorithm
```

23. Gegeben sei ***s*** eine Folge von natürlichen Zahlen dessen Elemente s_i der Form

$s_i = \begin{cases} x, falls\ i = 1 \\ x + 1, falls\ i = 2 \\ s_{(i-1)} @ s_{(i-2)} falls\ i > 2 \end{cases}$, (***i*** = 1, 2, …) sind. Der Operator @ verkettet die Ziffern des linken Operandes mit denen des rechten Operandes, in genau dieser Reihenfolge (diese Ziffern entsprechen der Darstellung in der 10-er Basis), und ***x*** ist eine natürliche Zahl ($1 \leq x \leq 99$). Zum Beispiel, falls ***x*** = 3, die Folge ***s*** wird die Werte 3, 4, 43, 434, 43443, ... enthalten. Geben Sie die Anzahl der Ziffern desjenigen Folgenglieds der Folge ***s,*** welches vor dem Glied, welches aus ***k*** ($1 \leq k \leq 30$) Ziffern gebildet ist vorkommt.

A. Falls ***x*** = 15 und ***k*** = 6, ist die Anzahl der Ziffern des Folgenglieds der Folge ***s,*** welches vor dem Glied, welches aus ***k*** Ziffern gebildet ist gleich mit 5.
B. Falls ***x*** = 2 und ***k*** = 8, ist die Anzahl der Ziffern des Folgenglieds der Folge ***s,*** welches vor dem Glied, welches aus ***k*** Ziffern gebildet ist gleich mit 5.
C. Falls ***x*** = 14 und ***k*** = 26, ist die Anzahl der Ziffern des Folgenglieds der Folge ***s,*** welches vor dem Glied, welches aus ***k*** Ziffern gebildet ist gleich mit 16.
D. Falls ***x*** = 5 und ***k*** = 13, ist die Anzahl der Ziffern des Folgenglieds der Folge ***s,*** welches vor dem Glied, welches aus ***k*** Ziffern gebildet ist gleich mit 10.

24. Gegeben sei eine Folge ***x*** bestehend aus ***n*** natürliche Zahlen ($3 \leq n \leq 10000$) und die natürliche Zahl ***k*** ($1 \leq k < n$). Der Unteralgorithmus `permCirc(n, k, x)` sollte die Kreispermutation der Folge ***x*** mit ***k*** Stellen nach links erzeugen. (Zum Beispiel, ist die Folge (4, 5, 2, 1, 3) eine Kreispermutation mit 2 Stellen nach links der Folge (1, 3, 4, 5, 2)). Leider ist der Unteralgorithmus `permCirc(n, k, x)` nicht korrekt, weil für manche Werte von ***n*** und ***k*** das Ergebnis nicht richtig ist.

```
Subalgorithm permCirc(n, k, x)
   c ← k
   For j = 1, c do
      permTo ← j
      nr ← x[permTo]
      For i = 1, n / c – 1 do
         permFrom ← permTo + k
         If permFrom > n then
            permFrom ← permFrom - n
         EndIf
         x[permTo] ← x[permFrom]
         permTo ← permFrom
      EndFor
      x[permTo] ← nr
   EndFor
EndSubalgorithm
```

Bestimmen Sie die Werte von ***n***, ***k*** und ***x,*** für die der Unteralgorithmus `permCirc(n, k, x)` eine Kreispermutation der Folge ***x*** mit ***k*** Stellen nach links erzeugt:

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

25. Eine natürliche, von Null verschiedene Zahl ***x*** heißt *glücklich*, wenn das Quadrat der Zahl als Summe von ***x*** aufeinanderfolgenden natürlichen Zahlen geschrieben werden kann. Zum Beispiel, 7 ist glücklich, weil $7^2 = 4 + 5 + 6 + 7 + 8 + 9 + 10$.

Welches der folgenden Unterprogramme überprüft, ob eine natürliche Zahl ***x*** ($2 \leq x \leq 1000$) *glücklich* ist? Jedes Unterprogramm hat als Eingabeparameter die Zahl ***x***, und als Ausgabeparameter die natürliche, von Null verschiedene Zahl ***start*** und die boolesche Variable ***istGlücklich.*** Falls die Zahl ***x*** glücklich ist, dann ***istGlücklich =*** *wahr* und ***start*** enthält die erste Zahl aus der Summe (z.B., falls ***x*** = 7, dann ***start*** = 4); falls die Zahl ***x*** nicht glücklich ist, dann ***istGlücklich =*** *falsch* und ***start*** hat den Wert -1.

A.

```
Subalgorithm glücklich(x, start, istGlücklich):
   xQuadrat ← x * x
   istGlücklich ← false
   start ← -1, k ← 1, s ← 0
   While k ≤ xQuadrat - x and not istGlücklich execute
      For i ← k, k + x - 1 execute
         s ← s + i
      EndFor
      If s = xQuadrat then
         istGlücklich ← true
         start ← k
      EndIf
   EndWhile
EndSubalgorithm
```

B.

```
Subalgorithm glücklich(x, start, istGlücklich):
 xQuadrat ← x * x
 istGlücklich ← false
 start ← -1, k ← 1
 While k ≤ xQuadrat - x and not istGlücklich execute
    s ← 0
    For i ← k, k + x - 1 execute
       s ← s + i
    EndFor
    If s = xQuadrat then
       istGlücklich ← true
       start ← k
    EndIf
    k ← k + 1
 EndWhile
EndSubalgorithm
```

C.

```
Subalgorithm glücklich(x, start, istGlücklich):
    If x MOD 2 = 0 then
       istGlücklich ← false
       start ← -1
    else
       istGlücklich ← true
       start ← (x + 1) DIV 2
    EndIf
EndSubalgorithm
```

D.

```
Subalgorithm glücklich(x, start, istGlücklich):
   If x MOD 2 = 0 then
      istGlücklich ← false
      start ← -1
   else
      istGlücklich ← true
      start ← x DIV 2
   EndIf
EndSubalgorithm
```

26. Gegeben sei der Unteralgorithmus `alg(x, b)` mit Eingabeparameter zwei von Null verschiedene natürliche Zahlen ***x*** und ***b*** ($1 \leq x \leq 1000$, $1 < b \leq 10$).

```
Subalgorithm alg(x, b):
   s ← 0
   While x > 0 do
      s ← s + x MOD b
      x ← x DIV b
   EndWhile
   return s MOD (b - 1) = 0
EndSubalgorithm
```

Was bewirkt dieser Unteralgorithmus?

A. Überprüft, ob die Summe der Ziffern der Darstellung von ***x*** in der Basis ***b*** - 1 durch ***b*** - 1 teilbar ist.
B. Überprüft, ob die natürliche Zahl ***x*** durch ***b*** - 1 teilbar ist.
C. Überprüft, ob die Summe der Darstellung von ***x*** in der Basis ***b*** durch ***b*** - 1 teilbar ist.
D. Überprüft, ob die Summer der Ziffern von ***x*** durch ***b*** - 1 teilbar ist.

27. Gegeben sei die Folge (1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, …), die folgendermaßen aufgebaut worden ist: beginnend mit der Folge der natürlichen Zahlen, werden die Zahlen, die nicht prim sind, mit deren echten Teilern ersetzt, jeder Teiler ***d*** einer Zahl kommt nur einmal vor. Welcher der folgenden Unteralgorithmen bestimmt das ***n***-te Element dieser Folge (***n*** ist eine natürliche Zahl, f1 $\leq n \leq$ 1000)?

A.

```
Subalgorithm identifizierung(n):
    a ← 1, b ← 1, c ← 1
    While c < n do
        a ← a + 1, b ← a, c ← c + 1, d ← 2
        f ← false
        While c ≤ n and d ≤ a DIV 2 do
            If a MOD d = 0 then
```

```
                c ← c + 1, b ← d, f ← true
            EndIf
            d ← d + 1
        EndWhile
        If f then
            c ← c - 1
        EndIf
    EndWhile
    return b
EndSubalgorithm
```

B.

```
Subalgorithm identifizierung(n):
    a ← 1, b ← 1, c ← 1
    While c < n do
        c ← c + 1, d ← 2
        While c ≤ n and d ≤ a DIV 2 do
            If a MOD d = 0 then
                c ← c + 1, b ← d
            EndIf
            d ← d + 1
         EndWhile
         a ← a + 1, b ← a
    EndWhile
    return b
EndSubalgorithm
```

C.

```
Subalgorithm identifizierung(n):
    a ← 1, b ← 1, c ← 1
    While c < n do
        a ← a + 1, d ← 2
        While c < n and d ≤ a do
            If a MOD d = 0 then
                c ← c + 1, b ← d
            EndIf
            d ← d + 1
        EndWhile
    EndWhile
    return b
EndSubalgorithm
```

D.

```
Subalgorithm identifizierung(n):
    a ← 1, b ← 1, c ← 1
    While c < n do
        b ← a, a ← a + 1, c ← c + 1, d ← 2
        While c ≤ n and d ≤ a DIV 2 do
            If a MOD d = 0 then
                c ← c + 1, b ← d
            EndIf
            d ← d + 1
        EndWhile
    EndWhile
    return b
EndSubalgorithm
```

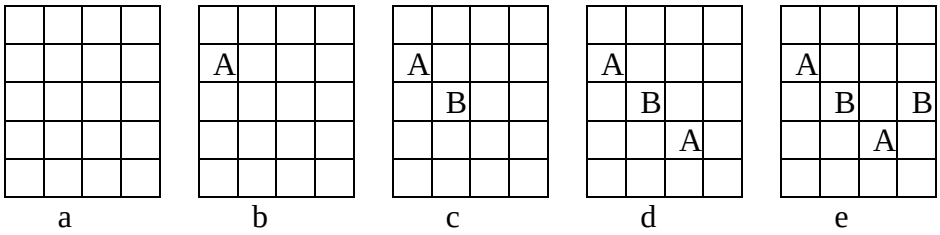
28. Das Rechteck dessen Seitenlängen ***m*** und ***n*** (***m***, ***n*** – natürliche Zahlen, $0 < \boldsymbol{m} < 101$, $0 < \boldsymbol{n} < 101$) sind, wird in Einheitsquadraten geteilt. Gegeben sei der Unteralgorithmus rechteck(m, n):

```
Subalgorithm rechteck(m, n)
   d ← m
   c ← n
   While d ≠ c do
      If d > c then
         d ← d – c
      else
         c ← c – d
      EndIf
   EndWhile
   return m + n – d
EndSubalgorithm
```

Geben Sie die Auswirkung dieses Unteralgorithmus.

A. Berechnet und gibt die Anzahl der Einheitsquadrate, die von einer Rechteckdiagonale durchquert sind an.
B. Bestimmt in ***d*** den größten gemeinsamen Teiler der Seiten des Rechtecks und gibt die Differenz der Summer der Rechteckseiten und ***d*** an.
C. Falls $\boldsymbol{m} = 8$ und $\boldsymbol{n} = 12$, gibt 16 zurück.
D. Falls $\boldsymbol{m} = 6$ und $\boldsymbol{n} = 11$, gibt 15 zurück.

29. Gegeben sei eine in $\boldsymbol{n} \times \boldsymbol{m}$ Felder (***n*** – Anzahl der Zeilen, ***m*** – Anzahl der Spalten, ***n***, ***m*** – natürliche Zahlen, $2 \leq \boldsymbol{n} \leq 100$, $2 \leq \boldsymbol{m} \leq 100$) eingeteilte rechteckige Tafel. Der Reihe nach werden zwei Spieler jeweils folgende Züge spielen: Bei jedem Zug wird ein Spieler ein einziges Feld schattieren und zwar dasjenige unschattierte Feld, welches auf der Diagonale mit dem beim vorherigen Zug von seinem Gegner schattiert wurde. Derjenige Spieler der kein Zug mehr spielen kann verliert. Der Spieler A spielt als erster indem er ein Feld schattiert.



Beispiel einer Spieltafel: a) Am Anfang ($n = 5$ și $m = 4$), b) Nach dem ersten Zug (A spielt), c) nach dem zweiten Zug (B spielt), d) nach dem dritten Zug (A spielt), e) nach dem vierten Zug (B spielt)

Bestimmen Sie in welchem Fall der Spieler A eine sichere Gewinnstrategie hat (d.h. er gewinnt das Spiel egal wie der Spieler B spielen wird) und welcher könnte der erste Zug von A sein, um das Spiel zu gewinnen.

A. Die Bedingung: ***m*** ist ungerade;
der erste Zug von A: ein Feld welches sich auf der ersten oberen Zeile der Tafel (Zeile 1) und auf einer Spalte mit ungeradem Index befindet.

B. Die Bedingung: ***n*** ist ungerade;
der erste Zug von A: ein Feld welches sich auf einer Zeile mit geradem Index und auf der ersten Spalte von links (Spalte 1) befindet;

C. Die Bedingung: ***n*** und ***m*** sind gerade;
der erste Zug von A: das Feld oben links (Zeile 1, Spalte 1);

D. Die Bedingung: mindestens eine der Zahlen ***n*** și ***m*** sind ungerade;
der erste Zug von A: das Feld oben links (Zeile 1, Spalte 1).

30. Eine Matrix mit 8 Zeilen deren Elemente ausschließlich 0 und 1 sind, hat folgende drei Eigenschaften:

a. Die erste Zeile enthält nur einmal die Zahl 1,
b. Die Zeile ***j*** enthält zweimal so viel von Null verschiedene Elemente als die Zeile $j-1$, für jedes $j \in \{2, 3, \ldots, 8\}$,
c. Die letzte Zeile enthält nur einmal die Zahl 0.

Welche ist die Anzahl der Vorkommnisse der Zahl 0 in der Matrix?

A. 777
B. 769
C. 528
D. eine solche Matrix gibt es nicht.

Richtige Antworten:

1. **D**
2. **B**
3. **A**
4. **B, C**
5. **C**
6. **A, C, D**
7. **B, C**
8. **A, B, C**
9. **A, B, D**
10. **A, C**
11. **A, D**
12. **B**
13. **B**
14. **C, D**
15. **A**
16. **B, D**
17. **A, C**
18. **A, B**
19. **A, C**
20. **A, D**
21. **B**
22. **A**
23. **B, C**
24. **A, D**
25. **B, C**
26. **B, C**
27. **A**
28. **A, C**
29. **A, D**
30. **A**