UNIVERSITÄT BABEŞ-BOLYAI
MATHEMATIK UND INFORMATIK FAKULTÄT

Aufnahmeprüfung Mate-Info - Modell

Schriftliche Prüfung in Informatik

Hinweise:

1. Man nimmt an, dass die Indizierung aller Arrays/Sequenzen bei 1 beginnt.
2. Bei jeder Ankreuzaufgabe (aus Teil A) ist wenigstens eine Antwort richtig, es können jedoch auch mehrere wahr sein. Diese müssen vom Kandidaten auf dem Püfungsblatt angegeben werden. Die der betreffenden Aufgabe entsprechenden Punkte werden genau dann vergeben, wenn alle richtigen Antworten und nur diese angegeben worden sind.
3. Bei den Aufgaben aus Teil B müssen vollständige Lösungen auf dem Püfungsblatt angegeben werden.
   - **a.** Die Lösungen müssen in *Pseudocode* oder *in einer Programmiersprache* (*Pascal/C/C++)* geschrieben werden.
   - **b.** Das erste Kriterium für die Auswertung der Lösungen ist die **Korrektheit** des Algorithmus, dann die **Performanz** bezüglich der *Ausführungszeit* und bezüglich des benutzten Speicherplatzes.
   - **c.** Es ist ***verpflichtend*** die (Unter-) Algorithmen vor der Lösungen ***zu beschreiben und zu erläutern***. Zusätzlich sollen Kommentare geschrieben werden, um das Verständnis der technischen Details der Lösung, der Bedeutung der Variablen, der Datenstrukturen usw. zu vereinfachen. Das Nichteinhalten dieser Voraussetzungen führt zum Verlust von 10% der Punkteanzahl der entsprechenden Aufgabe.
   - **d.** Es sollen keine vordefinierten Funktionen oder Bibliotheken benutzen werden (wie z.B.: STL, vordefinierte Funktionen für Zeichenfolgen/Strings, usw.).

Teil A (60 Punkte)

A.1. Welcher ist die Auswirkung des Unterprogramms? (6 Punkte)

Das Unterprogramm `generieren(n)` bearbeitet eine natürliche Zahl $\boldsymbol{n}$ ($0 < \boldsymbol{n} < 100$).

```
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 $\boldsymbol{n}$ (Summe der „Ziffern hoch 3") bis die Summe gleich ist mit der Zahl $\boldsymbol{n}$, und es gibt die Anzahl der ausgeführten Wiederholungen zurück.

B. es berechnet die Summe der Kubikzahlen der Ziffern der Zahl $\boldsymbol{n}$ und gibt diese Summe zurück

C. es berechnet die Summe der Kubikzahlen der Ziffern der Zahl $\boldsymbol{n}$, ersetzt die Zahl $\boldsymbol{n}$ mit der erhaltenen Summe und gibt diese Summe zurück

D. es berechnet die Anzahl der Ersetzungen von $\boldsymbol{n}$ mit der Summe der Kubikzahlen seiner Ziffern bis man einen Wert, der sich wiederholt, oder die Zahl selber erhält, und gibt diesen Wert zurück

A.2. Welche Werte werden benötigt? (6 Punkte)

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. $\boldsymbol{v}$ = (194, 121, 782, 0) und $\boldsymbol{k}$ = 4
B. $\boldsymbol{v}$ = (928) und $\boldsymbol{k}$ = 1
C. $\boldsymbol{v}$ = (9, 2, 8, 0) und $\boldsymbol{k}$ = 4
D. $\boldsymbol{v}$ = (8, 2, 9) und $\boldsymbol{k}$ = 3

A.3. Logische Auswertung (6 Punkte)

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

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

Bestimme wie oft das Unterprogramm `auswertung(s, k, i)` in der Ausführung 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 Male
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 Male
D. keinmal

A.4. Vereinigung (6 Punkte)

Das Unterprogramm `gehört(x, a, n)` überprüft, ob eine natürliche Zahl $\boldsymbol{x}$ zu der Menge $\boldsymbol{a}$ mit $\boldsymbol{n}$ Elementen gehört; $\boldsymbol{a}$ ist eine Sequenz mit $\boldsymbol{n}$ 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 $\boldsymbol{a}$, $\boldsymbol{b}$ und $\boldsymbol{c}$ Sequenzen sind, die Mengen von natürlichen Zahlen mit $\boldsymbol{n}$, $\boldsymbol{m}$ und beziehungweise $\boldsymbol{p}$ Elementen darstellen ($1 \leq \boldsymbol{n} \leq 200$, $1 \leq \boldsymbol{m} \leq 200$, $1 \leq \boldsymbol{p} \leq 400$). Die Eingabeparameter sind $\boldsymbol{a}$, $\boldsymbol{n}$, $\boldsymbol{b}$ und $\boldsymbol{m}$, und die Ausgabeparameter sind $\boldsymbol{c}$ und $\boldsymbol{p}$.

```
1.    Subalgorithm vereinigung(a, n, b, m, c, p):
2.      If n = 0 then
3.        For i ← 1, m execute
4.          p ← p + 1, cp ← bi
5.        EndFor
6.      else
7.        If not gehört(an, b, m) then
8.          p ← p + 1, cp ← an
9.        EndIf
10.       vereinigung(a, n - 1, b, m, c, p)
11.     EndIf
12.   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 $\boldsymbol{a}$ ein einziges Element enthält, dann entsteht in dem Unterprogramm `berechnung(a, n, b, m, c, p)` ein unendlicher Zyklus
B. wenn die Menge $\boldsymbol{a}$ 4 Elemente enthält, dann wird bei dem ersten Aufrufen des Unterprogramms `berechnung(a, n, b, m, c, p)` die Anweisung von der Linie 10 viermal ausgeführt
C. wenn die Menge $\boldsymbol{a}$ 5 Elemente enthält, dann wird bei dem ersten Aufrufen des Unterprogramms `berechnung(a, n, b, m, c, p)` die Anweisung von der Linie 2 fünfmal ausgeführt
D. wenn die Menge $\boldsymbol{a}$ die gleichen Elemente wie die Menge $\boldsymbol{b}$ enthält, dann wird die Menge $\boldsymbol{c}$ nach der Ausführung des Unterprogramms `berechnung(a, n, b, m, c, p)` dieselbe Anzahl von Elementen wie die Menge $\boldsymbol{a}$ haben

A.5. Potenzen (6 Punkte)

Welches der folgenden Unterprogramme berechnet korrekt den Wert a^b, wobei a und b zwei natürliche Zahlen sind ($1 \leq a \leq 11$, $0 \leq b \leq 11$).

A.
```
Subalgorithm potenz(a, b):
    ergebnis ← 1
    While b > 0 execute
        If b MOD 2 = 1 then
            ergebnis ← ergebnis * a
        EndIf
        b ←  b DIV 2
        a ← a * a
    EndWhile
    return ergebnis
EndSubalgorithm
```

B.
```
Subalgorithm potenz(a, b):
    If b ≠ 0 then
        If b MOD 2 = 1 then
            return potenz(a * a, b / 2) * a
        else
            return potenz(a * a, b / 2)
        EndIf
    else
        return 1
    EndIf
EndSubalgorithm
```

C.
```
Subalgorithm potenz(a, b):
    ergebnis ← 0
    While b > 0 execute
        ergebnis ← ergebnis * a
        b  ← b - 1
    EndWhile
    return rezultat
EndSubalgorithm
```

D.
```
Subalgorithm potenz(a, b):
    If b = 0 then
        return 1
    EndIF
    return a * potenz(a, b - 1)
EndSubalgorithm
```

A.6. Größtes Vielfaches (6 Punkte)

Welches der folgenden Unterprogramme gibt das größte Vielfache der natürlichen Zahl a an, Vielfaches, das kleiner oder gleich der natürlichen Zahl b ist ($0 < a < 10\,000$, $0 < b < 10\,000$, $a < b$)?

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
```

A.7. Datentyp (6 Punkte)

Ein Integer Datentyp repräsentiert auf x Bits (x ist eine natürliche streng positive Zahl) 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. $[-2^x, 2^x-1]$

A.8. Wie viele Male wird das Unterprogramm aufgerufen? (6 Punkte)

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 viele Male die Funktion *f* in der Ausführung der folgenden Anweisungsblock aufgerufen wurde:

```
a ← 4
b ← 3
c ← f(a, b)
```

A. 4 Male
B. 3 Male
C. Unendlich Male

D. keinmal

A.9. Zeichenfolgen (6 Punkte)

Seien alle Zeichenfolgen mit Länge $\boldsymbol{l} \in \{1, 2, 3\}$ bestehend aus den Buchstaben in 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

A.10. Positive Zahlen (6 Punkte)

Gegeben sei das Unterprogramm `positiveZahlen(m, a, n, b)`.

```
void positiveZahlen(int m,int a[], int &n,
int b[]){
    n = 0;
    for(int i = 0; i < n; i++){
      if (a[i] > 0){
         n = n + 1;
         b[n] = a[i];
      }
    }
}
```

```
procedure positiveZahlen(m:integer;
a:sir; var n:integer; var b:sir)
  Begin
    n := 0;
    for i := 1 to n do
      if (a[i] > 0) then
        begin
          n := n + 1;
          b[n] := a[i];
        end;
  End;
```

Welches ist das Ergebnis der Ausführung für den Aufruf `positiveZahlen(k, x, p, y)` mit k=4, $\boldsymbol{x}$=(-1,2,-3,4), $\boldsymbol{p}$ = -1 und dem leeren Array $\boldsymbol{y}$ = ().

A. $\boldsymbol{p}$ = 3 und $\boldsymbol{y}$ = (2, 4);
B. $\boldsymbol{p}$ = 0 und $\boldsymbol{y}$ = (2, 4);
C. $\boldsymbol{p}$ = 0 und $\boldsymbol{y}$ = ();
D. Hängt vom Wert von $\boldsymbol{k}$ ab.

<u>Teil B</u> (30 Punkte)

B.1. Magische Zahlen (15 Punkte)

Seien $\boldsymbol{p}$ und $\boldsymbol{q}$ zwei natürliche Zahlen ($2 \leq \boldsymbol{p} \leq 10$, $2 \leq \boldsymbol{q} \leq 10$). Eine natürliche Zahl heißt *magisch*, wenn die Menge der Ziffern, die benutzt wird um die Zahl in dem Zahlensystem mit Basis $\boldsymbol{p}$ darzustellen, identisch ist mit der Menge der Ziffern, die benutzt wird um die Zahl in dem Zahlensystem mit Basis $\boldsymbol{q}$ darzustellen. Zum Beispiel, für $\boldsymbol{p}$ = 9 und $\boldsymbol{q}$ = 7, ist $(31)_{10}$ eine *magische* Zahl, da $(34)_9 = (43)_7$, und für $\boldsymbol{p}$ = 3 und $\boldsymbol{q}$ = 9, ist $(9)_{10}$ eine magische Zahl, da $(100)_3 = (10)_9$. Sei das Unterprogramm `ziffernBasis(x, b, n, c)` für das Bestimmen der Ziffern der Zahl $\boldsymbol{x}$ zur Basis $\boldsymbol{b}$ (gespeichert in dem Array $\boldsymbol{c}$):

```
Subalgorithm ziffernBasis(x, b, c):
   While x > 0 execute
      c[x MOD b] ← 1
      x ← x DIV b
   EndWhile
SfSubalgorithm
```

Anforderungen:

a. Schreibe eine *rekursive* Variante (ohne Wiederholungsstrukturen) des Unterprogramms `ziffernBasis(x, b, c)` mit demselben Unterprogrammkopf wie das gegebene Unterprogramm, und welche die gleiche Auswirkung hat. (**5 Punkte**)
b. Gebe das mathematische Modell der rekursiven Variante des Unterprogramms `ziffernBasis(x, b, c)` (entwickelt bei a) an. (**3 Punkte**)
c. Schreibe ein Unterprogramm, das mithilfe des Unterprogramms `ziffernBasis(x, b, c)`, für zwei gegebene Basis $\boldsymbol{p}$ und $\boldsymbol{q}$ eine Sequenz $\boldsymbol{a}$ erstellt mit allen *magischen* Zahlen streng größer als 0 und streng kleiner als eine gegebene natürliche Zahl $\boldsymbol{n}$ ($1 < \boldsymbol{n} \leq 10\,000$). Die Eingabeparameter des

Unterprogramms sind ***p*** und ***q*** (die zwei Basis) und der Wert ***n***. Die Ausgabeparametern sind die Sequenz ***a*** und die Länge ***k*** der Sequenz ***a.*** (**7 Punkte**)

Beispiel: wenn ***p*** = 9, ***q*** = 7 și ***n*** = 500, dann wird die Sequenz ***a*** ***k*** = 11 Elemente haben: (1, 2, 3, 4, 5, 6, 31, 99, 198, 248, 297).

B.2. Schokoladenverkostung (15 Punkte)

Eine Werbefirma wirbt eine neue Schokoladensorte und plant Schokoladenproben zu ***n*** ($10 \leq n \leq 10000000$) Kindern, die in einem Kreis sitzen, zu verteilen. Die Angestellten der Firma merken, dass die Verteilung der Proben zu allen Kindern sehr teuer ist. Folglich, entscheiden sie Proben zu jedem ***k***-te ($0 < k < n$) Kind aus den ***n*** Kindern zu verteilen. Man zählt also die Kinder in Schritte mit dem Schritt ***k*** (wenn man zu dem letzten Kind ankommt, dann geht das Zählen weiter mit dem ersten Kind und so weiter). Beim Zählen werden alle Kinder berücksichtigt, egal ob das Kind Schokolade bekommen hat oder nicht. Das Zählen *hört genau dann auf wenn eine Schokoladenprobe zu einem Kind verteilt werden muss, das schon Schokolade bekommen hat.*

Anforderungen:

a. Bestimme die Eigenschaft, welche die Position k eines Kindes erfüllen muss, damit das ***k***-te Kind Schokolade kriegt. Begründe die Antwort. (**3 Punkte**)
b. Erkläre (in eueren Worten und mithilfe von mathematischen Formeln) welche die Anzahl der Kinder ist, welche Schokoladenprobe erhalten. Begründe die Antwort. (**2 Punkte**)
c. Schreibe ein Unterprogramm, das die Anzahl der Kinder (***nr***) bestimmt, welche keine Schokoladenprobe erhalten. Die Eingabeparameter des Unterprogramms sind die natürliche Zahlen ***n*** und ***k***, und der Rückgabewert (Ausgabeparameter) wird die natürliche Zahl ***nr*** sein. (**10 Punkte**)

Beispiel 1: wenn ***n*** = 12 und ***k*** = 9, dann ***nr*** = 8 (das erste, zweite, vierte, fünfte, siebente, achte, zehnte und elfte Kind erhalten keine Schokolade).
Beispiel 2: wenn ***n*** = 15 und ***k*** = 7, dann ***nr*** = 0 (alle Kinder erhalten Schokolade).

Bemerkung:

1. Alle Aufgaben sind verpflichtend.
1. Schmierblätter werden nicht berücksichtigt.
2. Die Bewertung beginnt bei 10 Punkten.
3. Die Bearbeitungszeit beträgt 3 Stunden.

PUNKTEANZAHL & LÖSUNGEN

ANFANGSPUNKTEANZAHL ..... 10 Punkte

TEIL A ..... 60 Punkte

A. 1. Welcher ist die Auswirkung des Unterprogramms? Antwort D ..... 6 Punkte
A. 2. Welche Werte werden benötigt? Antwort A, C ..... 6 Punkte
A. 3. Logische Auswertung. Antwort B ..... 6 Punkte
A. 4. Vereinigung. Antwort B, D ..... 6 Punkte
A. 5. Potenzen. Antwort A, B, D ..... 6 Punkte
A. 6. Größtes Vielfaches. Antwort C, D ..... 6 Punkte
A. 7. Datentyp. Antwort B, C ..... 6 Punkte
A. 8. Wie viele Male wird das Unterprogramm aufgerufen? Antwort C ..... 6 Punkte
A. 9. Zeichenfolgen. Antwort A ..... 6 Punkte
A. 10. Positive Zahlen. Antwort C ..... 6 Punkte

TEIL B ..... 30 Punkte

B. 1. Magische Zahlen ..... 15 Punkte

B.1.a. rekurisve Version des Unterprogramms `ziffernBasis(x, b, c)` **..... 5 Punkte**

– derselbe Unterprogrammkopf ..... 1 Punkt
– Bedingung für die Aufhaltung der Rekursivität ..... 1 Punkt
– Selbstaufruf (Logik, Parameter) ..... 2 Punkte
– zurückgegebene Werte ..... 1 Punkt

```
Subalgorithm ziffernBasis(x, b, c):
   If x > 0 then
      c[x MOD b] ← 1
      ziffernBasis(x DIV b, b, c)
   EndIf
EndSubalgorithm
```

B.1.b. mathematisches Modell für `ziffernBasis(x, b, c)` **..... 3 Punkte**

$$ziffernBasis(x,b,c) = \begin{cases} -, & wenn\ x = 0 \\ ziffernBasis(x\ DIV\ b, b, c'), wobei\ c'_{x\ MOD\ b} = 1, & ansonsten \end{cases}$$

B.1.c. Unterprogramm für die Erhaltung der Sequenz a ..... 7 Punkte

– Die Überprüfung der Eigenschaft einer *magischen Zahl*

- V1: mithilfe der Identität der zwei characteristischen Vektoren für die Mengen der Ziffern der Zahl in den zwei Zahlensystemen (mit Basis ***p***, beziehungsweise Basis ***q)*** .....5 Punkte

```
// man erstellt den charakteristischen Vektor der Ziffern der Zahl x zur Basis p
// man bestimmt der Reihe nach die Ziffern der Zahl x zur Basis q
//      wenn die aktuelle Ziffer nicht in der Zahlendarstellung zur Basis p vorkommt, dann ist
//            x nicht magisch
//      ansonsten wird der entsprechende Wert in dem Ziffernvektor inkrementiert
// wenn in dem Ziffernvektor ein Wert 1 vorhanden ist, dann heißt das, dass die entsprechenden
// Ziffern in der Zahlendarstellung zur Basis p, aber nicht in der Zahlendarstellung zur Basis
// q vorkommen. Folglich ist die Zahl nicht magisch.
bool magischeZahl(int x, int p, int q){
        // überprüft ob x magisch ist in Bezug auf Basis p und q
        int ziffern[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        ziffernBasis(x, p, cifre);
        while (x != 0){     // man bestimmt die Menge der Ziffern der Zahl x zur Basis q
                int lz = x % q;  //lz – letzte Ziffer
                if (cifre[lz] == 0)  // wenn die aktuelle Ziffer (zur Basis q) nicht in der
                                     // Zahlendarstellung zur Basis p vorkommt
                        return false;
                ziffern[lz]++;
                x = x / q;
        }
        for (int i = 0; i < 10; i++){
                if (ziffern[i] == 1)   //wenn die Ziffer i in der Zahlendarstellung zur Bais p, aber
                                       // nicht zur Basis q verwendet wird
                        return false;
        }
        return true;
}
```

- V2: andere Algorithmus-Varianten, die korrekt sind, aber nicht so effizient ....maximal 3 Punkte

– Der Aufbau der Sequenz ***a*** .....2 Punkte

```
void sequenzMagischeZahlen(int p, int q, int n, int &k, int seq[]){
        k = 0;
        for (int i = 1; i < n; i++){
                if (magischeZahl(i, p, q))
                        seq[k++] = i;
        }
}
```

B. 2. Schokoladenverkostung..... 15 Punkte

B.2.a. Man kann das Zählen im Kreis als lineares Zählen in mehreren kleinen Sequenzen betrachten, wobei jede Sequenz n Kinder enthält. Dann kriegt man am Ende eine lange Sequenz mit p Kinder, wobei p Vielfache von n ist. Das Zählen wird beendet, wenn das n-te Kind (aus einer kleinen Sequenz) Schokolade erhaltet (so ist das nächste Kind, das Schokolade erhalten muss, das k-te Kind aus der nächsten Sequenz), Zum Beispiel, für ***n*** = 12 Kinder und ***k*** = 9, bildet man mehrere kleine Sequenzen mit je ***n*** Kinder:

Kind Index	1	2	3	4	5	6	7	8	9	10	11	12
Mit/ohne Schokolade	0	0	0	0	0	0	0	0	0	0	0	0

Index große Sequenz	1	2	3	4	5	6	7	8	**9**	10	11	12
Kind Index	1	2	3	4	5	6	7	8	**9**	10	11	12
Mit/ohne Schokolade	0	0	0	0	0	0	0	0	**1**	0	0	0

Index große Sequenz	13	14	15	16	17	**18**	19	20	21	22	23	24
Kind Index	1	2	3	4	5	**6**	7	8	9	10	11	12
Mit/ohne Schokolade	0	0	0	0	0	**1**	0	0	1	0	0	0

Index große Sequenz	25	26	**27**	28	29	30	31	32	33	34	35	**36**
Kind Index	1	2	**3**	4	5	6	7	8	9	10	11	**12**
Mit/ohne Schokolade	0	0	**1**	0	0	1	0	0	1	0	0	**0**

Hier hört die Schokoladen Verteilung auf, da als nächstes eine Schokoladenprobe zu einem Kind verteilt werden muss, das schon Schokolade bekommen hat.

Index große Sequenz	37	38	39	40	41	42	43	44	**45**	46	47	48
Kind Index	1	2	3	4	5	6	7	8	**9**	10	11	12
Mit/ohne Schokolade	0	0	1	0	0	1	0	0	**1**	0	0	1

Die Kinder, die Schokolade erhalten haben sind genau die, deren Index in der großen Sequenz Vielfache von ***k*** ist. ..... **2 Punkte**

B.2.b. Die Anzahl der Kinder, die Schokolade erhalten: ***p*** muss auch Vielfache von ***k*** sein, also ***p*** = cmmmc(***n***, ***k***). Aus den p Kinder, haben genau ***p*** / ***k*** Kinder Schokolade erhalten, also die Anzahl der Kinder, die keine Schokolade erhalten haben, ist: ***n*** - ***p*** / ***k*** = ***n*** - cmmmc(***n***, ***k***) / ***k*** = ***n*** - (***n*** * ***k*** / cmmdc(***n***, ***k***)) / ***k*** = ***n*** - ***n*** / cmmdc(***n***, ***k***)..... **3 Punkte**

B.2.c. Entwicklung des Unterprogramms..... **10 Punkte**

- V1: die korrekte Bestimmung des Wertes ***nr*** (mit der Formel ***nr*** = ***n*** – ***n***/*cmmdc*(***n***, ***k***)...10 Punkte

```
// berechnet und gibt zurück das cmmdc (kleinste gemeinsame Vielfache) zweier
// natürlichen Zahlen a und b
int cmmdc(int a, int b){
        if ((a == b) && (a == 0))
                return 1;
        if (a * b == 0)
                return a + b;
        while (b != 0){
                int c = b;
                b = a % b;
                a = c;
        }       //while
        return a;
}

int schokolandenverkostung(int n, int k){
```

```
        return n - n / cmmdc(n, k);
}
```

- V2: die korrekte Bestimmung des Wertes ***nr*** (durch Simulation, zirkuläre Liste) .....7 Punkte

```
int schokolade(int n, int k){
        const int MAXLEN = 10000000;
        bool schokolade[MAXLEN];
        for (int i = 1; i <= n; i++)
                schokolade[i] = false;
        int i = k;
        int anzahlKinderOhneSchokolade = n;
        while (!schokolade[i]){ //Simulation zirkuläre Liste
                schokolade[i] = true;
                anzahlKinderOhneSchokolade--;
                i += k;
                if (i > n){
                        i -= n;
                }
        }
        return anzahlKinderOhneSchokolade;
}
```