BABEŞ-BOLYAI UNIVERSITÄT
FAKULTÄT FÜR MATHEMATIK UND INFORMATIK

Aufnahmeprüfung - Variante 2

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 (30 Punkte)

A.1. Welche ist die Auswirkung des Unterprogramms? (5 Punkte)

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 ist mit der Zahl n, 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 wiederholt die Summe der Kubikzahlen der Ziffern der Zahl n bis sich eine Summe wiederholt und es gibt die Anzahl der ausgeführten Wiederholungen zurück.

E. 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.

A.2. Welche Werte werden benötigt? (5 Punkte)

Sei das Unterprogramm `verarbeitung(v, k)`, wobei v eine Sequenz mit k natürlichen Zahlen ist ($1 \leq k \leq 1\,000$).

```
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

E. v = (912, 0, 120, 8, 0) und k = 5

A.3. Logiche Auswertung (5 Punkte)

Sei ***s*** eine Sequenz von ***k*** Elementen vom booleschen Typ 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 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

E. unendlich oft

A.4. Vereinigung (5 Punkte)

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 ($1 \leq n \leq 200$, $1 \leq x \leq 1000$). 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 ($1 \leq n \leq 200$, $1 \leq m \leq 200$, $1 \leq p \leq 400$). 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 gleiche Anzahl von Elementen wie die Menge ***b*** hat, 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

E. 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

A.5. Potenzen (5 Punkte)

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

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 ← 1
   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
    aux ← potenz(a, b DIV 2)
    If b MOD 2 = 0 then
       return aux * aux
    else
       return a * aux * aux
    EndIf
EndSubalgorithm
```

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

A.6. Größtes Vielfaches (5 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
```

E.
```
Subalgoritgm f(a, b):
  c ← a
  While c < b execute
    c ← c + a
  EndWhile
  If c = b then
    return c
  else
    return c – a
  EndIf
EndSubalgorithm
```

Teil B (60 Punkte)

B.1. Polynomauswertung (10 Punkte)

Sei das Unterprogramm `auswertung(n, koeff, x)`, wobei ***koeff*** ein Vektor mit $n + 1$ Elementen ist, welche reelle Zahlen aus dem Intervall [-100, 100] sind und welche die Koeffizienten eines Polynoms vom Grad n darstellen, $P(x) = koeff_1 * x^n + koeff_2 * x^{n-1} + ... + koeff_n * x + koeff_{n+1}$, in absteigender Reihenfolge der Potenzen von x (n ist eine natürliche Zahl, $1 \le n \le 10$). Das Unterprogramm bestimmt den Wert des Polynoms für einen gegebenen Wert x (x ist eine reelle Zahl aus dem Intervall [-10, 10]).

```
Subalgorithm auswertung(n, koeff, x):
    val ← 0.0
    For i ← 1, n + 1 execute
       val ← val * x + koeff[i]
    EndFor
    return val
EndSubalgorithm
```

Schreibe eine *rekursive* Variante (die keine Schleifen enthält) des Unterprogramms `auswertung(n, koeff, x)` mit dem gleichen Unterprogrammkopf und der gleichen Auswirkung wie das gegebene Unterprogramm.

B.2. Durchschnitt (25 Punkte)

Seien zwei Sequenzen, die beide *unterschiedliche* ganze Zahlen zwischen -30 000 und 30 000 enthalten. Sequenz ***a*** enthält ***n*** ($0 < n \leq 10\,000$) Elemente. Sequenz ***b*** enthält ***m*** ($0 < m \leq 10\,000$) Elemente *geordnet in steigender Reihenfolge*.

Schreibe ein Unterprogramm, das die Sequenz ***c*** mit ***k*** ($0 \leq k \leq 10\,000$) Elementen bestimmt, wobei ***c*** alle *gemeinsame* Elemente der zwei Sequenzen enthält, nur einmal genommen, in jedwelcher Reihenfolge. Die Eingabeparameter des Unterprogramms sind die zwei Sequenzen (***a*** und ***b***) und deren Längen (***n*** und ***m***). Die Ausgabeparameter sind die Sequenz ***c*** und deren Länge ***k***. Falls es keine gemeinsame Elemente gibt, dann wird ***k*** den Wert 0 haben.

Beispiel*:** wenn $n = 4$ und $a = (5, -7, -2, 3)$, $m = 5$ und $b = (-2, 3, 5, 7, 8)$, dann enthält die Sequenz ***c genau $k = 3$ Elemente und $c = (5, -2, 3)$.

B.3. Zahlensequenz ohne Brüder (25 Punkte)

Sei eine Sequenz ***a***, die ***n*** ($0 < n \leq 10\,000$) Elemente enthält, welche unterschiedliche natürliche Zahlen verschieden von Null und kleiner als 30 000 sind. Zwei Zahlen heißen *Brüder* wenn diese *unterschiedlich sind* und wenn sie *wenigstens zwei unterschiedliche gemeinsame Ziffern enthalten*. Zum Beispiel, 5867 und 17526 sind *Brüder*, aber 5867 und 152 sind nicht *Brüder*. 131 und 114 sind auch keine Brüder.

Anforderungen:

i. Schreibe ein Unterprogramm, das überprüft ob zwei natürliche Zahlen ***a*** und ***b*** ($0 < a \leq 30\,000$, $0 < b \leq 30\,000$) Brüder sind. Die Eingabeparameter sind die zwei Zahlen ***a*** und ***b***. Der Ausgabeparameter ist ***sindBrüder*** und er enthält den Wahrheitswert *wahr* falls ***a*** und ***b*** Brüder sind, und *falsch* ansonsten. **(11 Punkte)**

ii. Schreibe ein Unterprogramm, das die längste Untersequenz der Sequenz **a** bestimmt, die aus Elementen besteht, die *kein Bruder* in der Sequenz ***a*** haben. Eine Untersequenz besteht aus Elemente der Sequenz die sich auf aufeinanderfolgende Positionen befinden. Die Eingabeparameter sind die Sequenz ***a*** und ihre Länge ***n***. Die Ausgabeparameter sind die Anfangsposition der Untersequenz ***start*** und ihre Länge ***k***. Falls es mehrere Untersequenzen mit derselben maximalen Länge gibt, dann soll die letzte ausgegeben werden. Falls es keine solche Untersequenz gibt, dann wird ***start*** den Wert -1 und ***k*** den Wert 0 haben. **(14 Punkte)**

Beispiel: Sei $n = 11$ und $a = (12345, 9, 100, 567, 5678, 345, 123, 8989, 222, 11, 78)$. Die Zahlen ohne Brüder aus der Sequenz ***x*** sind: 9, 100, 8989, 222, 11, und die gesuchte Untersequenz ist (8989, 222, 11), also ***start*** = 8 und $k = 3$.

Bemerkung:

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

BAREM

OFICIU ..... 10 puncte

Partea A ..... 30 puncte
A. 1. Oare ce face? Răspunsul E..... 5 puncte
A. 2. Ce valori sunt necesare? Răspunsurile A, C ..... 5 puncte
A. 3. Evaluare logică. Răspunsul B ..... 5 puncte
A. 4 Reuniune. Răspunsurile B, E ..... 5 puncte
A. 5. Exponențiere. Răspunsurile A, B, C, D, E ..... 5 puncte
A. 6. Cel mai mare multiplu. Răspunsurile C, D, E..... 5 puncte

Partea B ..... 60 puncte
B. 1. Evaluare polinom ..... 10 puncte
- respectarea parametrilor de intrare și ieșire..... 2 puncte
- condiția de oprire din recursivitate..... 2 puncte
- autoapel ..... 2 puncte
- valoarea returnată la oprirea recursivității..... 2 puncte
- valoarea returnată la continuarea recursivității ..... 2 puncte

B. 2. Intersecție ..... 25 puncte
- respectarea parametrilor de intrare și ieșire..... 2 puncte
    - variante:
        - folosirea căutării binare ..... 23 puncte
        - fără căutare binară ..... maxim 18 puncte

B. 3. Secvență de numere fără frați..... 25 puncte
- respectarea parametrilor de intrare și ieșire..... 2 puncte
- proprietatea de frate..... 10 puncte
- determinarea unei secvențe ..... 9 puncte
- determinarea celei mai lungi secvențe..... 4 puncte