UNIVERSITÄT BABEŞ-BOLYAI
MATHEMATIK UND INFORMATIK FAKULTÄT

Wettbewerb 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? (6p)

Sei ein Unterprogramm `alg(x, b)` mit Eingabeparametern zwei natürliche Zahlen ***x*** und ***b*** ($1 \leq x \leq 1000$, $1 < b \leq 10$).

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

Welche ist die Auswirkung des Unterprogramms?

A. berechnet die Summe der Ziffern der Darstellung der natürlichen Zahl ***x*** zur Basis ***b***
B. überprüft ob die Summe der Ziffern der Darstellung der natürlichen Zahl ***x*** zur Basis ***b*** - 1 durch ***b*** - 1 teilbar ist
C. überprüft ob die natürliche Zahl ***x*** durch ***b*** - 1 teilbar ist
D. überprüft ob die Summe der Ziffern der Darstellung der natürlichen Zahl ***x*** zur Basis ***b*** durch ***b*** - 1 teilbar ist

A.2. Was wird ausgedruckt? (6p)

Sei das folgende Programm:

Varianta C++/C

```
int sum(int n, int a[], int s){
   s = 0;
   int i = 1;
   while(i <= n){
      if(a[i] != 0) s += a[i];
      ++i;
   }
   return s;
}


int main(){
   int n = 3, p = 0, a[10];
   a[1] = -1; a[2] = 0; a[3] = 3;
   int s = sum(n, a, p);
   cout << s << ";" << p; // printf("%d;%d", s, p);
   return 0;
}
```

Varianta Pascal

```
type vector = array[1..10] of integer;

function sum(n:integer; a:vector; s:integer):integer;
  var i : integer;
  begin
     s := 0; i := 1;
     while (i <= n) do
        begin
           if (a[i] <> 0) then s := s + a[i];
           i := i + 1;
        end;
     sum := s;
  end;
var n, p, s : integer;
          a : vector;
begin
    n := 3; a[1] := -1; a[2] := 0; a[3] := 3; p := 0;
    s := sum(n, a, p);
    writeln(s, ';', p);
end.
```

Welches ist das Ergebnis, das nach der Ausführung des Programms, ausgedruckt wird?

A. `0;0`
B. `2;0`
C. `2;2`
D. Keine Antwort ist richtig

A.3. Logischer Ausdruck (6p)

Sei der folgende logische Ausdruck `(X OR Z) AND (NOT X OR Y)`. Wähle die Werte für ***X***, ***Y***, ***Z***, sodass das Auswertungsergebnis des Ausdruckes `WAHR` ist:

A. ***X*** ← FALSCH; ***Y*** ← FALSCH; ***Z*** ← WAHR;
B. ***X*** ← WAHR; ***Y*** ← FALSCH; ***Z*** ← FALSCH;
C. ***X*** ← FALSCH; ***Y*** ← WAHR; ***Z*** ← FALSCH;
D. ***X*** ← WAHR; ***Y*** ← WAHR; ***Z*** ← WAHR;

A.4. Berechnung (6p)

Sei das Unterprogramm `berechnung(a, b)` mit Eingabeparametern zwei natürliche Zahlen ***a*** și ***b***, $1 \leq a \leq 1000$, $1 \leq b \leq 1000$.

```
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 folgenden Aussagen sind falsch?

A. wenn ***a*** und ***b*** gleich sind, dann gibt das Unterprogramm den Wert von ***a*** zurück
B. wenn ***a*** = 1000 und ***b*** = 2, dann ruft sich das Unterprogramm 10 Male selbst auf
C. der Wert, der von dem Unterprogramm berechnet und zurückgegeben wird, ist ***a*** / 2 + 2 * ***b***
D. die Anweisung auf der Linie 5 wird nie ausgeführt

A.5. Element Identifizierung (6p)

Sei die Sequenz (1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, ...) gebildet folgendermaßen: anfangend von der Sequenz der natürlichen Zahlen, wird jede Zahl, die nicht prim ist, mit ihrem echten Teiler ersetzt, wobei jeder Teiler ***d*** nur einmal für jede Zahl betrachtet wird. Welches der folgenden Unterprogramme bestimmt das ***n***-te Element dieser Sequenz (***n*** – natürliche Zahl, $1 \leq n \leq 1000$)?

A.
```
Subalgorithm finde(n):
   a ← 1, b ← 1, c ← 1
   While c < n execute
      a ← a + 1, b ← a, c ← c + 1, d ← 2
      f ← false
      While c ≤ n and d ≤ a DIV 2 execute
         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 finde(n):
   a ← 1, b ← 1, c ← 1
   While c < n execute
      c ← c + 1, d ← 2
      While c ≤ n and d ≤ a DIV 2 execute
         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 finde(n):
   a ← 1, b ← 1, c ← 1
   While c < n execute
      a ← a + 1, d ← 2
      While c < n and d ≤ a execute
         If a MOD d = 0 then
            c ← c + 1, b ← d
         EndIf
         d ← d + 1
      SfCâtTimp
   SfCâtTimp
   return b
EndSubalgorithm
```

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

A.6. Primfaktoren (6p)

Sei das Unterprogramm `primfaktoren(n, d, k, x)`, das die ***k*** Primfaktoren einer natürlichen Zahl ***n*** bestimmt, wobei die Suche der Primfaktoren vom Wert ***d*** anfängt. Die Eingabeparameter sind die natürliche Zahlen ***n***, ***d*** und ***k***, und die Ausgabeparameter sind die Sequenz ***x*** mit den ***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 execute
```

```
      n ← n DIV d
   EndWhile
   If n > 1 then
      primfaktoren(n, d + 1, k, x)
   EndIf
EndSubalgorithm
```

Bestimme wie oft das Unterprogramm `primfaktoren(n, d, k, x)` in der Ausführung der folgenden Anweisungssequenz sich selbst aufruft:

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

A. 3 Male
B. 5 Male
C. 9 Male
D. genau so oft wie in der folgenden Anweisungssequenz:

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

A.7. Welcher ist die Auswirkung des Unterprogramms? (6p)

Sei das 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. $E(n) = 1* 2 + 2 * 3 + 3 * 4 + ... + (-1)^{n} * n * (n + 1)$

A.8. Logischer Ausdruck (6p)

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*** ← *FALSCH*; ***Z*** ← *WAHR*;
C. ***X*** ← *FALSCH*; ***Y*** ← *WAHR*; ***Z*** ← *FALSCH*;
D. ***X*** ← *WAHR*; ***Y*** ← *FALSCH*; ***Z*** ← *WAHR*;

A.9. Werte des Eingabeparameters (6p)

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

A.10. Fehlende Anweisungen (6p)

Gegeben sei das folgenden Unterprogramm:

```
1:    Subalgorithm suche(x, n, val):
2:          If n = 1 then
3:                return (x[1] = val)
4:          else
5:                return suche(x, n - 1, val)
6:          EndIf
7:    EndSubalgorithm
```

Welche Anweisung oder Anweisungen müssen hinzugefügt werden und wo genau, sodass das Unterprogramm `suche(x, n, val)` bestimmt, ob der Wert ***val*** in dem Array ***x*** mit ***n*** Elementen enthalten ist (***n*** natürliche Zahl strikt größer als 0)?

A. Linie 5 muss folgendermaßen geändert werden: `return ((x[n] = val) and suche(x - 1, n, val))`
B. Linie 5 muss folgendermaßen geändert werden: `return ((x[n] = val) or suche(x, n - 1, val))`
C. Linie 5 muss folgendermaßen geändert werden: `if (x[n] = val) then return true else return suche(x, n - 1, val)`
D. Keine Anweisung muss geändert werden

Teil B (30 Punkte)

1. Präfix (15 Punkte)

Die Kontrollziffer einer natürlichen Zahl wird als Summe der Ziffern der Zahl, dann Summe der Ziffern der Summe, usw. berechnet, bis die erhaltene Summe nur eine Ziffer enthält. Zum Beispiel, ist die *Kontrollziffer* der Zahl 182 genau 2 (1 + 8 + 2 = 11, 1 + 1 = 2).

Eine Zahl ***p***, die genau ***k*** Ziffer enthält ist ein *Präfix* der Zahl ***q*** mit wenigstens ***k*** Ziffern, falls die Zahl gebildet aus den ersten ***k*** Ziffern der Zahl ***q*** (von links nach rechts) gleich ist mit ***p***. Zum Beispiel, ist 17 ein Präfix von 174, und 1713 ein Präfix von 1 713 242.

Sei die natürliche Zahl ***nr*** ($0 < nr \leq 30\,000$) und eine Matrix (zweidimensionales Array) ***A*** mit ***m*** Zeilen und ***n*** Spalten ($0 < m \leq 100$, $0 < n \leq 100$), die als Elemente natürliche Zahlen kleiner als 30 000 hat. Sei das Unterprogramm `kontrollziffer(x)` für die Bestimmung der Kontrollziffer der Zahl ***x***:

```
Subalgorithm kontrollziffer(x):
    s ← 0
   While x > 0 execute
      s ← s + x MOD 10
      x ← x DIV 10
      If x = 0 then
         If s < 10 then
            Return s
         else
            x ← s
            s ← 0
         EndIf
      EndIf
   EndWhile
   return s
EndSubalgorithm
```

Anforderungen:

a. Schreibe eine *rekursive* Variante (ohne Wiederholungsstrukturen) des Unterprogramms `kontrollziffer(x)` 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 `kontrollziffer(x)` (entwickelt bei a) an. (**3 Punkte**)
c. Schreibe ein Unterprogamm, welches das Unterprogramm `kontrollziffer(x)` benutzt, um das längste Präfix (benotet ***präfix***) der Zahl ***nr*** zu bestimmen, das mit den Kontrollziffern der Elementen aus dem gegebenen Matrix gebildet werden kann. Eine solche Kontrollziffer kann beliebig viele Male bei dem Erstellen des Präfixes benutzt werden. Falls es keinen solche Präfix gibt, dann wird ***präfix*** gleich -1 sein. Die Eingabeparameter des Unterprogramms sind die Zahlen ***nr***, ***m***, ***n***,die Matrix ***A***, und der Ausgabeparameter ist ***präfix***. (**7 Punkte**)

Beispiel*:** für ***nr = 12319, ***m*** = 3 und ***n*** = 4 und die Matrix $A = \begin{pmatrix} 182 & 12 & 274 & 22 \\ 22 & 1 & 98 & 56 \\ 5 & 301 & 51 & 94 \end{pmatrix}$,

ist das längste Präfix ***prefix*** = 1231, wobei die Kontrollziffern folgende Werte haben:

Matrixelement	182	12	274	22	1	98	56	5	301	51	94
Kontrollziffer	2	3	4	4	1	8	2	5	4	6	4

2. Robi-Garten (15 Punkte)

Ein Gärtner mit einer Leidenschaft für Technologie entscheidet sich, eine „Armee" von Roboter für das Gießen der Beete in seinem Garten zu benutzen. Er möchte Wasser von der Wasserquelle am Ende des Hauptgangs, der den Garten durchquert, benutzen. Jedem Beet wird ein Roboter zugeordnet und jeder Roboter muss ein einziges Beet gießen. Alle Roboter starten die Beet-Bewässerungs-Aktion am Morgen um dieselbe Uhrzeit (zum Beispiel 5:00:00) von der Wasserquelle und arbeiten zugleich und ununterbrochen in demselben Zeitintervall. Sie gehen auf dem Hauptweg bis zu ihrem Beet, gießen das Beet und gehen dann zurück zu der Wasserquelle, um den Wassertank nachzufüllen. Am Ende des Aktivitätszeitintervalls hören alle Roboter zugleich auf, unabhängig von ihren aktuellen Zuständen. Ursprünglich gibt es bei der Wasserquelle einen einzigen Leitungshahn. Der Gärtner merkt aber, dass es Verzögerungen in dem Gießen-Programm gibt, da die Roboter der Reihe nach warten müssen, um die Wassertanks nachzufüllen, also denkt er, die beste Lösung sei, mehrere Leitungshähne für die Wasserversorgnung der Roboter zu installieren. Die Roboter fangen morgens mit vollen Wassertänken an. Zwei Roboter können nicht gleichzeitig denselben Leitungshahn benutzen.

Man kennt: das Zeitintervall t (ausgedrückt in Sekunden)**,** in welchem die n Roboter arbeiten, die Anzahl der Sekunden d_i, die gebraucht werden, um die Strecke zwischen der Wasserquelle und dem entsprechenden Beet zurückzulegen, die Anzahl der Sekunden u_i, die gebraucht werden, um das entsprechende Beet zu gießen, und die Tatsache, dass die Nachfüllung des eigenen Wassertanks genau eine Sekunde für jeden Roboter dauert (t, n, d_i, u_i – natürliche Zahlen, $1 \le t \le 20000$, $1 \le n \le 100$, $1 \le d_i \le 1000$, $1 \le u_i \le 1000$, $i = 1, 2, ..., n$).

Anforderungen:

a. Gebe die Roboter an, die sich bei der Wasserquelle zu einem bestimmten Zeitpunkt mt ($1 \le mt \le t$) treffen. Begründe die Antwort.
Bemerkung: die Roboter werden durch ihre Nummer identifiziert. (**3 Punkte**)

b. Welche ist die minimale Anzahl von zusätzlichen Wasserhähnen ***minZusätzHähne***, die der Gärtner installieren muss, damit die Roboter für die Nachfüllung des Wassertanks überhaupt nicht aufeinander warten müssen? Begründe die Antwort. (**2 Punkte**)

c. Schreibe ein Unterprogramm, das die minimale Anzahl der zusätzlichen Wasserhähne ***minZusätzHähne*** berechnet. Die Eingabeparameter sind die Zahlen n und t, die Sequenzen d und u mit je n Elementen, und der Ausgabeparameter ist ***minZusätzHähne***. (**10 Punkte**)

Beispiel 1: wenn $t = 32$, $n = 5$, $d = (1, 2, 1, 2, 1)$, $u = (1, 3, 2, 1, 3)$, dann ***minZusätzHähne*** = 3. Erklärung: der Roboter, der das Beet 1 gießen muss braucht eine Sekunde, um beim Beet anzukommen, eine Sekunde, um das Beet zu gießen und noch eine Sekunde, um zu der Wasserquelle zurückzugehen; er soll also seinen Wassertank nach 1 + 1 + 1 = 3 Sekunden nach der Startzeit (5:00:00) nachfüllen, also um 5:00:03 Uhr; er füllt seinen Wassertank in einer Sekunde nach und kehrt zu dem Beet um 5:00:04 Uhr zurück; er kommt zurück um den Wassertank nachzufüllen um 5:00:07 Uhr, um dann wieder das Beet zu gießen, usw.; also der erste, der zweite, der vierte und der fünfte Roboter treffen sich bei der Wasserquelle um 05:00:23 Uhr; folglich, braucht man 3 zusätzliche Wasserhähne.

Beispiel 2: wenn $t = 37$, $n = 3$, $d = (1, 2, 1)$, $u = (1, 3, 2)$, dann ***minZusätzHähne*** = 1.

Bemerkung:

1. Alle Aufgaben sind verpflichtend.
1. Schmierblätter werden nicht berücksichtigt.
2. Die Bewertung beginnt bei 10 Punkten.
3. Die Bearbeitungszeit beträgt 3.5 Stunden.

PUNKTEANZAHL & LÖSUNGEN
ANFANGSPUNKTEANZAHL ..... 10 Punkte

Teil A ..... 60 Punkte
A. 1. Welcher ist die Auswirkung des Unterprogramms? Antwort C, D ..... **6 Punkte**
A. 2. Was wird ausgedruckt? Antwort B ..... **6 Punkte**
A. 3. Logischer Ausdruck. Antwort A, D ..... **6 Punkte**
A. 4. Berechnung. Antwort A, C, D ..... **6 Punkte**
A. 5. Element Identifizierung. Antwort A ..... **6 Punkte**
A. 6. Primfaktoren. Antwort A, D ..... **6 Punkte**
A. 7. Welcher ist die Auswirkung des Unterprogramms? Antwort A ..... **6 Punkte**
A. 8. Logischer Ausdruck. Antwort A, B, D ..... **6 Punkte**
A.9. Werte des Eingabeparameters. Antwort A, B, D ..... **6 Punkte**
A.10. Fehlende Anweisungen. Antwort B, C ..... **6 Punkte**

Teil B ..... 30 Punkte
B. 1. Präfix ..... 15 Punkte
B.1.a. rekurisve Version des Unterprogramms `kontrollziffer(x)` ..... **5 Punkte**
– derselbe Unterprogrammkopf ..... 1 Punkt
– Bedingung für die Aufhaltung der Rekursivität ..... 1 Punkt
– Selbstaufruf (Logik, Parameter) ..... 2 Punkte
– zurückgegebene Werte ..... 1 Punkt

```
Subalgorithm kontrollziffer(x):
   If x > 9 then
      s ← x MOD 10 + kontrollziffer(x DIV 10)
      If s < 10 then
         Return s
      else
         Return cifraControl(s)
      EndIf
   else
      Return x
   EndIf
EndSubalgorithm
```

B.1.b. mathematisches Modell für `kontrollziffer(x)` ..... **3 Punkte**

$$kontrollziffer(x) = \begin{cases} x, & wenn\ x < 10 \\ kontrollziffer(x\ MOD\ 10) + kontrollziffer(x\ DIV\ 10)), & ansonsten \end{cases}$$

B.1.c. Unterprogramm für das längste Präfix ..... 7 Punkte

Variante 1: die Matrix wird nur einmal durchlaufen und man bildet den charakteristischen Vektor der Ziffern für die entsprechenden Kontrolziffern aus der Matrix. Man bildet ein Array mit den Ziffern der angegebenen Zahl (*nr*). Das Array dieser Ziffern wird durchlaufen (anfangend mit der meisten signifikante Ziffer), indem man dabei das Vorkommen der Ziffer in den charakteristischen Vektor überprüft ..... **7 Punkte**

Bemerkung: dieselbe Idee kann auch mithilfe eines Frequenzvektors (anstatt des charackteristischen Vektors) gelöst werden

```
const int ANZAHLMAXZIFFERN = 10;
int präfixMaxZiffern(int nr, int m, int n, int A[][MAX]){
   bool vorkommen[ANZAHLMAXZIFFERN];   int ziffern[MAX];
    int anzahlZiffern; int i = 0;
   for (i = 0; i < ANZAHLMAXZIFFERN; i++)     // Initialisierung des characteristischen Vektors
      vorkommen[i] = 0;
   for (i = 0; i < m; i++){
      for (int j = 0; j < n; j++){
         vorkommen[kontrollziffer(A[i][j])] = 1;            // das Vorkommen der Kontrollziffer
      }
   }
   anzahlZiffern = 0;                                  // Anzahl der Ziffern der angegebenen Zahl
   while (nr > 0){
```

```
      ziffern[anzahlZiffern++] = nr % 10;// Elemente des Arrays mit den Ziffern der angegebenen Zahl
      nr /= 10;
   }
   i = anzahlZiffern - 1;                              // das Array der Ziffern wird durchlaufen
   while ((i >= 0) && (vorkommen[ziffern[i]]))// gibt es eine Kontrollziffer = aktuelle Ziffer aus nr?
      i--;
   return anzahlZiffern - i - 1;                                  // die Länge des längsten Präfixes
}
```

<u>Variante 2:</u> Man bildet ein Array mit den Ziffern der angegebenen Zahl (*nr*). Man sucht jede Ziffer der Zahl in der Matrix (die Matrix wird also mehrmals durchlaufen) ..... **5 Punkte**

```
bool sucheZiffer(int ziffer, int m, int n, int A[][MAX]){
   for(int i = 0; i < m; i++){
      for(int j = 0; j < n; j++){
         if(ziffer == kontrollziffer(A[i][j]))         // die gesuchte Ziffer wird in dem Matrix
            return true;                                        // der Kontrollziffern gefunden
      }
   }
   return false;                                          // man hat die Ziffer nicht gefunden
}
int päfixMaxZiffern_v2(int nr, int m, int n, int A[][MAX]){
   int ziffern[MAX];
   int anzahlZiffern = 0;                                 // Anzahl der Ziffern der angegebenen Zahl
   while(nr > 0){
      ziffern[anzahlZiffern++] = nr % 10; // Elemente des Arrays mit den Ziffern der angegebenen Zahl
      nr /= 10;
   }
   int i = anzahlZiffern - 1;                             // das Array der Ziffern wird durchlaufen
   int p = 0;
   while(i >= 0){
      if(sucheZiffer(ziffern[i], m, n, A)){   // wir suchen die Ziffern der Zahl in dem Matrix A
         p++;
         i--;
      }
      else
         return p;
   }
   return p;     // die Anzahl der aufeinanderfolgenden Ziffern aus der nr gefunden in dem Matrix
(Länge des Präfixes)
}
```

B. 2. Robi-Garten ..... **15 Punkte**

Wenn $n = 5$, $d = (1, 2, 1, 2, 1)$, $u = (1, 3, 2, 1, 3)$, $t = 32$, dann berechnet man die Werte q:
2 * 1 + 1 + 1 = 4, 2* 2 + 3 + 1 = 8, 2 * 1 + 2 + 1 = 5, 2 * 2 + 1 + 1 = 6, 2 * 1 + 3 + 1 = 6 => (4, 8, 5, 6, 6)
Man bildet den Array v mit $t = 32$ Nullwerte. Man nimmt der Reihe nach den Wert q für jeden Roboter und man inkrementiert die Werte, welche einem Vielfache von q entsprechen, in dem Array.

$q = 4$

	1	2	3	**4**	5	6	7	**8**	9	10	11	**12**	13	14	15	**16**	17	18	19	**20**
v	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**

	21	22	23	**24**	25	26	27	**28**	29	30	31	**32**
v	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**

$q = 8$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	0	0	0	1	0	0	0	2	0	0	0	1	0	0	0	2	0	0	0	1

	21	22	23	24	25	26	27	28	29	30	31	32
v	0	0	0	2	0	0	0	1	0	0	0	2

$q = 5$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	0	0	0	1	1	0	0	2	0	1	0	1	0	0	1	2	0	0	0	2

	21	22	23	24	25	26	27	28	29	30	31	32
v	0	0	0	2	1	0	0	1	0	1	0	2

$q = 6$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

v	0	0	0	1	1	1	0	2	0	1	0	2	0	0	1	2	0	1	0	2

	21	22	23	24	25	26	27	28	29	30	31	32
v	0	0	0	3	1	0	0	1	0	2	0	2

q = 6

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	0	0	0	1	1	2	0	2	0	1	0	3	0	0	1	2	0	2	0	2

	21	22	23	24	25	26	27	28	29	30	31	32
v	0	0	0	4	1	0	0	1	0	3	0	2

Man bestimmt den maximalen Wert aus dem Array ***v***. In diesem Beispiel in das maximale Wert 4 (an dem Mmoment 24), also man braucht 4 - 1 = 3 zusätzliche Wasserhähne.

B.2.a. an einem bestimmten Zeitpunkt ***mt*** ($1 \leq mt \leq t$) treffen sich an der Quelle die Roboter mit dem Wert ***q*** (gleich mit der Summe zwischen der Zeit, die erforderlich ist, um zu dem Beet und zurück zu gelangen, der Zeit, die erforderlich ist, um das Beet zu gießen und der Zeit, die erforderlich ist, um den Wassertank nachzufüllen) Vielfache von ***mt*** ..... 3 Punkte

B.2.b. minimale Anzal von zusätzlichen Wasserhähnen ist gleich mit dem maximalen Wert aus dem Array ***v*** - 1, wobei das Array für jede Zeitpunkt die Anzahl der Roboter, die sich an der Quelle treffen, speichert .....2 Punkte

B.2.c. Entwicklung des Unterprogramms

- V1: man benutzt einen Frequenzvektor für die Vielfache der Arbeitszeiten jedes Roboters .....10 Punkte
  - c.1. Einhaltung der Eingabe- und Ausgabeparamters ..... **2 Punkte**
  - c.2. Berechnung der Arbeitszeit (***q*** = 2 * Reise + Gießen + Nachfüllen).....**1 Punkt**
  - c.3. Bearbeitung des Frequenzvektors.....**4 Punkte**
    - c.3.1. Initialisierung des Vektors .....1 Punkt
    - c.3.2. Aktualisierung der Frequenz.....3 Punkte
      1. v1a oder v1b: durchlaufe alle q Werte.....3 Punkte
      2. v2a oder v2b: durchlaufe jeden einzelnen Wert und überprüfe ob es einen Vielfache von q ist .....2 Punkte
  - c.4. Bestimmung der maximalen Anzahl von Wasserhähnen .....2 **Punkte**
      3. gleichzeitig mit der Inkrementierung oder nach der Beendung der Inkrementierung.....1 Punkt
  - c.5. Bestimmung der Anzahl der zusätzlichen Wasserhähnen (max - 1) .....**1 Punkt**

```
int robiV1a(int n, int d[], int u[] int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
        aux[i] = 0;
    for (int j = 1; j <= n; j++){
        int q = d[j] * 2 + u[j] + 1;
        for (int i = q; i <= t; i = i + q){
                aux[i]++;
                if (max < aux[i])
                        max = aux[i];
        }       //for i
    }   //for j
    return max - 1;
}
```

```
int robiV1b(int n, int d[], int u[], int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
        aux[i] = 0;
    for (int j = 1; j <= n; j++){
        int q = d[j] * 2 + u[j] + 1;
        for (int i = q; i <= t; i = i + q){
                aux[i]++;
        }       //for i
    }   //for j
    for (int i = 1; i <= t; i++){
        if (max < aux[i])
                max = aux[i];
    }   //for i
    return max - 1;
}
```

```
int robiV2a(int n, int d[], int u[], int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
        aux[i] = 0;
    for (int j = 1; j <= n; j++){
        int q = d[j] * 2 + u[j] + 1;
        for (int i = q; i <= t; i = i + 1){
                if (i % q == 0){
                        aux[i]++;
                        if (max < aux[i])
                                max = aux[i];
                }       //if (i % q)
        }       //for i
    }   //for j
```

```
int robiV2b(int n, int d[], int u[], int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
        aux[i] = 0;
    for (int j = 1; j <= n; j++){
        int q = d[j] * 2 + u[j] + 1;
        for (int i = q; i <= t; i = i + 1){
                if (i % q == 0){
                        aux[i]++;
                }       //if (i % q)
        }       //for i
    }   //for j
    for (int i = 1; i <= t; i++){
        if (max < aux[i])
```

```
    return max - 1;
}
```

```
                max = aux[i];
    }    //for i
    return max - 1;
}
```

- V2: Simulieren.....8 Punkte
  - c.1. Einhaltung der Eingabe- und Ausgabeparamters.....2 Punkte
  - c.2. Berechnung der Arbeitszeit (***q*** = 2 * Reise + Gießen + Nachfüllen).....1 Punkte
  - c.3. Wiederholungsschleife für die Zeit.....0.5 Punkte
  - c.4. Wiederholungsschleife für die Roboter.....0.5 Punkte
  - c.5. Bestimmung der erforderlichen Wasserhähnen an einem bestimmten Zeitpunkt.....1 Punkt
  - c.6 Bestimmung der maximalen Anzahl von Wasserhähnen.....2 Punkte
  - c.7. Bestimmung der Anzahl der zusätzlichen Wasserhähnen.....1 Punkt

```
int robiV3(int n, int d[], int u[], int t){
    int minAnzahlHähne = 1;
    int aktuelleZeit = 1;
    while (aktuelleZeit <= t){
         int aktuelleAnzahlHähne = 0;
         for (int j = 1; j <= n; j++){
                 int q = 2 * d[j] + u[j] + 1;
                 if (aktuelleZeit % q == 0) //wenn sich der i-te Roboter zu der
                 aktuelleAnzahlHähne++;    //aktuellen Zeitpunkt bei der Quelle befindet
         }
         if (aktuelleAnzahlHähne > minAnzahlHähne)
                 minAnzahlHähne = aktuelleAnzahlHähne;
         aktuelleZeit++;
    }
    return minAnzahlHähne - 1;
}
```