BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Admission Exam September 2018

Written Exam for Computer Science

Important observations for candidates:

1. All arrays are indexed starting from 1.
2. Multiple answer problem statements (Part A) can have one or more correct answers. Candidates must write the answers on their contest sheet (not the sheets with the problem statements). Solutions are graded only if they identify all the correct answers, and only them.
3. Complete solutions are expected for problem statements in Part B.
   - **a.** Solutions will be described using *pseudocode* or a *programming language* (*Pascal/C/C++)*.
   - **b.** The first criterion when evaluating solutions is algorithm ***correctness***, followed by its ***performance*** regarding *time of execution* and *required memory space*.
   - **c.** ***Providing description and justification*** for all (sub)algorithms before their implementation is ***mandatory***. In addition, ***comments*** will be used to describe details of the technical implementation, meaning of used identifiers, data structures and so on. Failure to observe these requirements leads to a 10% penalty from the problem score.
   - **d.** Using predefined functions or libraries is prohibited (for example: *STL*, predefined string functions).

Part A (30 points)

A.1. What does the following subalgorithm do? (5p)

The `generate(n)` subalgorithm processes natural number n ($0 < n < 100$).

```
Subalgorithm generate(n):
    nr ← 0
    For i ← 1, 1801 do
        used_i ← false
    EndFor
    While not used_n do
        sum ← 0, used_n ← true
        While (n ≠ 0) do
            digit ← n MOD 10, n ← n DIV 10
            sum ← sum + digit * digit * digit
        EndWhile
        n ← sum, nr ← nr + 1
    EndWhile
    return nr
EndSubalgorithm
```

State the effect of this subalgorithm.

A. repeatedly calculates the sum of the cubes of n's digits until the sum is equal to n, after which it returns the number of repetitions
B. calculates the sum of the cubes of n's digits and returns this sum
C. calculates the sum of the cubes of n's digits, replaces n with the obtained sum and returns it
D. repeatedly calculates the sum of the cubes of n's digits until a sum is obtained for the second time and returns the number of repetitions
E. calculates how many times number n was replaced with the sum of the cubes of n's digits until a previously calculated value or the number itself is obtained; this number is then returned

A.2. What values are required? (5p)

Consider subalgorithm `process(v, k)`, where v is an array of k natural numbers ($1 \leq k \leq 1\,000$).

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

State for which values of v and k the subalgorithm returns 928.

A. v = (194, 121, 782, 0) and k = 4
B. v = (928) and k = 1
C. v = (9, 2, 8, 0) and k = 4
D. v = (8, 2, 9) and k = 3
E. v = (912, 0, 120, 8, 0) and k = 5

A.3. Logical Evaluation (5p)

Let us consider array ***s*** with ***k*** boolean elements and subalgorithm `evaluation(s, k, i)`, where ***k*** and ***i*** are natural numbers ($0 \leq i \leq k \leq 100$).

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

State how many times subalgorithm `evaluation(s, k, i)` is called itself given the following instruction sequence:

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

A. 3 times
B. the same number of times as in the following instruction sequence

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

C. 6 times
D. never
E. infinite number of times

A.4. Reunion (5p)

Consider subalgorithm `belongs(x, a, n)`, which checks whether natural number ***x*** belongs to set ***a*** having ***n*** elements; ***a*** is an array of ***n*** elements that represents a natural numbers set ($1 \leq n \leq 200$, $1 \leq x \leq 1000$). Let us consider subalgorithms `reunion(a, n, b, m, c, p)` and `compute(a, n, b, m, c, p)`, described below, where ***a***, ***b*** and ***c*** are arrays that represent natural number sets having ***n***, ***m*** and ***p*** elements respectively ($1 \leq n \leq 200$, $1 \leq m \leq 200$, $1 \leq p \leq 400$). Input parameters are ***a***, ***n***, ***b, m*** and ***p***, and output parameters are ***c*** and ***p***.

```
1.      Subalgorithm reunion(a, n, b, m, c, p):
2.        If n = 0 then
3.          For i ← 1, m do
4.            p ← p + 1
5.            cp ← bi
6.          EndFor
7.        else
8.          If not belongs(an, b, m) then
9.            p ← p + 1
10.           cp ← an
11.         EndIf
12.         reunion(a, n - 1, b, m, c, p)
13.       EndIf
14.     EndSubalgorithm
```

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

Which of the following statements are always true:

A. when set ***a*** contains a single element, calling subalgorithm `compute(a, n, b, m, c, p)` results in an infinite loop
B. when set ***a*** contains 4 elements, calling subalgorithm `compute(a, n, b, m, c, p)` results in executing the instruction on line 12 a number of 4 times
C. when set ***a*** contains 5 elements, calling subalgorithm `compute(a, n, b, m, c, p)` results in executing the instruction on line 2 of the `reunion` subalgorithm a number of 5 times
D. when sets ***a*** and ***b*** have the same number of elements, after the execution of subalgorithm `compute(a, n, b, m, c, p)` set ***c*** will have the same number of elements as set ***a***
E. when sets ***a*** and ***b*** have the same elements, after the execution of subalgorithm `compute(a, n, b, m, c, p)` set ***c*** will have the same number of elements as set ***a***

A.5. Exponentiation (5p)

Which of the following algorithms correctly computes a^b, where a and b are natural numbers ($1 \le a \le 11$, $0 \le b \le 11$).

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

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

C.
```
Subalgorithm expo(a, b):
    result ← 1
    While b > 0 do
        result ← result * a
        b  ← b - 1
    EndWhile
    return result
EndSubalgorithm
```

D.
```
Subalgorithm expo(a, b):
    If b = 0 then
        return 1
    EndIf
    aux ← expo(a, b DIV 2)
    If b MOD 2 = 0 then
        return aux * aux
    else
        return a * aux * aux
    EndIf
EndSubalgorithm
```

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

A.6. Largest multiple (5p)

Which of the following subalgorithms returns the largest multiple of number a, multiple that is smaller or equal with natural number b ($0 < a < 10\,000$, $0 < b < 10\,000$, $a < b$)?

A.
```
Subalgorithm f(a, b):
  c ← b
  While c MOD a = 0 do
    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.
```
Subalgorithm f(a, b):
  c ← a
  While c < b do
    c ← c + a
  EndWhile
  If c = b then
    return c
  else
    return c – a
  EndIf
EndSubalgorithm
```

Part B (60 points)

B.1. Polynomial evaluation (10 points)

Consider the `evaluation(n, coef, x)` subalgorithm, where ***coef*** is a vector containing $n + 1$ real numbers in interval [-100, 100] which represent the coefficients of the n degree polynome $P(x) = coef_1 * x^n + coef_2 * x^{n-1} + ... + coef_n * x + coef_{n+1}$, provided in decreasing order of the powers of x (n is a natural number, $1 \le n \le 10$). The subalgorithm calculates the polynome's value at point x (x is a real number in interval [-10, 10]).

```
Subalgorithm evaluation(n, coef, x):
    val ← 0.0
    For i ← 1, n + 1 do
        val ← val * x + coef[i]
    EndFor
    return val
EndSubalgorithm
```

Write a *recursive* implementation without loops for subalgorithm `evaluation(n, coef, x)`, which has the same signature and effect.

B.2. Intersection (25 points)

Let us consider two arrays, each containing *distinct* integer numbers between -30 000 and 30 000. Array ***a*** has ***n*** ($0 < n \leq 10000$) elements, and array ***b*** has ***m*** ($0 < m \leq 10000$) elements *sorted in ascending order*.

Write a subalgorithm that determines array ***c*** having ***k*** ($0 \leq k \leq 10000$) elements formed using all the *common* elements of the two arrays, each of them appearing a single time, in any order. The input parameters are the two arrays (***a*** and ***b***) and their lengths (***n*** and ***m***). The array ***c*** and its length ***k*** are the output parameters. If there are no common elements, then ***k*** is 0.

Example: if ***n*** = 4, ***a*** = (5, -7,-2, 3), ***m*** = 5 and ***b*** = (-2, 3, 5, 7, 8), array ***c*** has ***k*** = 3 elements and ***c*** = (5, -2, 3).

B.3. Brotherless number sequence (25 points)

Let us consider array ***x*** with ***n*** ($0 < n \leq 10\,000$) *distinct*, non-zero natural numbers smaller than 30 000. Two numbers are called *brothers* if they *are distinct* and if they have *at least two distinct digits in common*. For example, 5867 and 17526 are *brothers*, but 5867 and 152 are not *brothers*. Also, 131 and 114 are not brothers.

Requirements:

i. Write a subalgorithm that checks whether natural numbers ***a*** and ***b*** are brothers ($0 < a \leq 30\,000$, $0 < b \leq 30\,000$). Input parameters are the two numbers ***a*** and ***b***. The output parameter ***isBrother*** will be *true* if ***a*** and ***b*** are brothers, and *false* otherwise.

ii. Write a subalgorithm that determines the longest subarray of ***x*** that consists of elements that *have no brother* in array ***x***. A subarray of an array is made up of array's elements in consecutive positions. Input parameters are array ***x*** and its length ***n***. Output parameters are the starting index of the subarray ***start*** and its length ***k***. In case multiple subarrays of the same length exist, consider the last one of them. If no such subarray exists, then ***start*** will be -1 and ***k*** will be 0.

Example: If ***n*** = 11 and ***x*** = (12345, 9, 100, 567, 5678, 345, 123, 8989, 222, 11, 78). The brotherless numbers contained in ***x*** are 9, 100, 8989, 222, 11, and the longest subarray is (8989, 222, 11), so ***start*** = 8 and ***k*** = 3.

Note:

1. All subjects are mandatory.
2. Drafts are not taken into consideration for grading.
3. Default 10 points.
4. Working time is 3 hours.

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