Algoritmi care lucreaza pe numere (fara tablouri si date structurate)

15 noiembrie 2025

conf.dr. Adrian Sterca

1. Se considera algoritmul ***factor(n)***, unde ***n*** este un numar natural pozitiv:

```
Algorithm factor(n)
        a ← 0
        For  i ← 5, n;  i ← i * 5  execute
                a ← a  +  [n/i]
        EndFor
        Return a
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?
A. Pentru orice n $\geq$ 625, algoritmul va returna o valoare mai mare decat suma cifrelor lui **n**
B. Pentru n = 24 si n = 25, algoritmul va returna aceeasi valoare
C. Pentru n = 125, valoarea returnata va fi 31
D. Rezultatul algoritmului reprezinta numarul maxim de factori primi ai lui **n** mai mari decat 5

Raspuns: A si C
a > n/5 > 2n/10 > 2*10^(nr.cifre n -1) > (nr. cifre n) * 9

2. Se considera subprogramele ***ceFace1(a, b)*** si ***ceFace2(a, b)*** unde **a** si **b** sunt doua numere naturale, cu proprietatea ca **a** $\leq$ **20**, **b** $\leq$ **12**. Operatia **<<= k** reprezinta operatia de shiftare la stanga a variabilei cu **k biti**, iar **&** reprezinta operatia **AND** (bitwise).

```
Algorithm ceFace1(a, b)
    If  b = 0  then
        Return 1
    EndIf
    If  b  MOD  2 =  1  then
        Return  a ∗ ceFace1 (a, b-1 )
    EndIf
    s ⟵ ceFace1 (a,  b DIV 2 )
    Return s ∗ s
EndAlgorithm
```

```
Algorithm ceFace2(a, b)
    s ⟵ 1
    For   i ⟵ 1, b, i<<=1  execute
        If  (b  &  i)  then
            s ⟵ s ∗ a
        EndIf
        a ⟵ a ∗ a
    EndFor
    Return s
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?
A. Apelul **ceFace1(4, 4)** va returna valoarea 256
B. Apelul **ceFace1(a, b)** va returna mereu aceeasi valoarea ca apelul **ceFace2(a,b)**
C. Algoritmul **ceFace2(a, b)** este mai eficient din punct de vedere al timpului de executie decat algoritmul **ceFace1(a, b)**
D. Algoritmul **ceFace2(a, b)** este mai eficient din punct de vedere al memoriei utilizate decat algoritmul **ceFace1(a, b)**

Raspuns: A, B, C
Programul calculeaza **exponentul a^b**.

3. Se considera urmatoarea expresie logica:
((A **XOR** B) **AND** (C **XOR** D)) **OR** (**NOT** (A **AND** B) **XOR** (C **OR** D))

Precizati pentru ce valori ale lui A,B,C,D, expresia are valoarea True.
A. A = False, B = False, C = True, D = False
B. A = True, B = False, C = True, D = True
C. A = True, B = True, C = True, D = True
D. A = True, B = True, C = True, D = False

Raspuns: C si D (a doua subexpresie e True)

4. Se considera algoritmul ***ceFace(x)*** definit alaturat, unde ***x*** si ***c*** sunt numere naturale, cel mult 10^9.

```
Algorithm ceFace(x)
   If x < 10 then
        If  x  MOD  2  =  0  then
              Returneaza x
        Else
              Returneaza -1
        EndIf
   Else
        If  x  MOD  2  =  0 then
              c  =  ceFace( x DIV 10 )
              If  c = −1   OR   c > x MOD 10  then
                    Returneaza  x  MOD  10
              Else
                    Returneaza c
              EndIf
         Else
              Returneaza ceFace( x  DIV  10 )
        EndIf
   EndIf
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?
A. Algoritmul returneaza cea mai mare cifra para a numarului x
B. Pentru x = 719480888, apelul ceFace(x) va returna valoarea 8
C. Pentru x = 11317915, apelul ceFace(x) va returna valoarea −1
D. Complexitatea timp a algoritmului este O(log x)

Raspuns: C si D.
Algoritmul calculeaza recursiv cea mai mica cifra para a numarului x si o returneaza, iar in cazul numerelor care nu contin cifre pare, algoritmul returneaza −1. Deoarece in baza 10, un numar x are aprox. $\log_{10} x$ cifre, complexitatea algoritmului este O(log x).

5. Se considera subalgoritmul ***ceFace(n)***, unde ***n*** este un numar natural, $1 \leq \boldsymbol{n} \leq 1000000000$. S-au folosit notația ***x%y*** pentru restul împărțirii numărului întreg *x* la numărul întreg *y* și ***[x]*** pentru partea întreagă a numărului real *x.*

```
Algorithm ceFace(n)
      If ([n / 10]) == 0)
            If (n % 2 == 0)
                  return n;
```

```
            EnfIf
            return 0;
      Else
            If (n % 10 % 2 = 1)
                  return ceFace([n / 10]);
            EndIf
            return ceFace([n / 10]) * 10 + n % 10;
      EndIf
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?

A. La apelul ceFace(2528) se va returna 2528
B. La apelul ceFace(335) se va returna 0
C. Cand algoritmul este apelat pentru n – numar impar, se va returna 0
D. Cand algoritmul este apelat pentru n – numar par, se va returna n

De câte ori se apelează funcția pentru a = 253401976?

a. De 3 ori
b. De 9 ori
c. De 6 ori
d. De 7 ori

Raspuns: B si b

Explicatie: Algoritmul elimina dintr-un numar toate cifrele impare, iar daca toate sunt impare, se returneaza 0. Pentru numarul a = 253401976 algoritmul se apeleaza de 9 ori fiindca numarul are 9 cifre.

6. Se da subalgoritmul ceFace(n), unde n este un numar natural nenul.

```
Algorithm ceFace(n)
      If   n = 0  then
            Return 1
      Else  If   n = 1   then
            Return 2
      Else
            Return   ceFace(n - 1)  +  ceFace(n - 2)
      EndIf
EndAlgorithm
```

Ce calculeaza acest subalgoritm?

A. Numarul de siruri binare de lungime n care nu contin doua zerouri consecutive
B. Numarul de siruri binare de lungime n care nu contin doi de 1 consecutivi
C. Al n-lea termen din sirul lui Fibonacci
D. Numarul de permutari ale unei multimi cu n elemente

Raspuns: A, B.
Subalgoritmul implementeaza recurenta f(n) = f(n − 1) + f(n − 2) cu conditiile initiale f(0) = 1, f(1) = 2. Aceasta corespunde numarului de siruri binare de lungime n fara doi de 1 consecutive (optiunea B). Analog si Optiunea A este adevarata, C este incorecta ($F_0 = 0 \neq 1$), iar D nu are legatura cu recurenta.

7. Se considera algoritmul ***ceFace(x, y)***, unde ***x*** si ***y*** sunt numere naturale ($0 \leq x, y \leq 103$).

```
Algorithm ceFace(x, y)
        If   x > y   then
                x ⟵ x − y
                y ⟵ y + x
                x ⟵ y – x
        EndIf
        z ⟵ 0
        For   i ⟵ x, y   execute
                If   i mod 3 = 0   and   i mod 2 ≠ 0   then
                        z ⟵ z + i
                EndIf
        EndFor
        Return z
EndAlgorithm
```

Care dintre urmatoarele situatii algoritmul returneaza valoarea 24?
A. x = 4, y = 19
B. x = 15, y = 8
C. x = 6, y = 14
D. x = 20, y = 9

Raspuns: A, B, D
Algoritmul calculeaza suma tuturor numerelor din intervalul [x, y], daca $x \leq y$, sau din intervalul [y, x], daca $y < x$, care sunt divizibile cu 3 si nu sunt divizibile cu 2.

8. Se considera algoritmul ***frozen(n)***, unde ***n*** este un numar natural nenul ($1 \leq n \leq 10^2$).

```
Algorithm frozen(n)
    If  n < 2 then
```

```
        Return
    EndIf
    For   i ← n, 2,−3  execute
        frozen (n - 1)
        Write i − 2
        frozen ((i + 1) DIV 2)
        Write n + 3
    EndFor
EndAlgorithm
```

In urma apelului ***frozen(6)*** se afiseaza un sir de cifre. Care sunt cifrele de pe pozitiile 16 - 20, prima pozitie fiind numerotata cu 1?

A. 10568
B. 80510
C. 62057
D. 56805

Raspuns: Primele 20 de cifre afisate sunt: 05105620573051056805, iar cifrele de pe pozitiile 16-20 sunt 56805 (optiunea D).

Apeluri functie frozen() si tipariri in ordine temporala (ordinea temporala a apelurilor este ordinea verticala; cand un apel e generat din interiorul altui apel, el este identat la dreapta fata de primul apel; de exemplu, mai jos, apelul frozen(4) este generat din interiorul apelului frozen(5), frozen(3) este generat din interiorul apelului frozen(4)):

```
frozen(5)
        frozen(4)
                frozen(3)
                        frozen(2)
                                frozen(1)
                                Write 0
                                frozen(1)
                                Write 5
                        Write 1
                        frozen(2)
                                frozen(1)
                                Write 0
                                frozen(1)
                                Write 5
                        Write 6
                Write 2
                frozen(2)
                        frozen(1)
                        Write 0
```

```
                    frozen(1)
                    Write 5
               Write 7
          Write 3
          frozen(3)
          .....
          Write 8
          frozen(4)
          .....
          Write 0
          frozen(1)
Write 4
fronzen(3)
....
Write 9
frozen(5)
.....
Write 1
frozen(2)
....
```

9. Se considera algoritmul ***ceFace(a, b)***, unde ***a*** si ***b*** sunt numere naturale nenule ($1 \leq a \leq 102$, $0 \leq b \leq 25$).

```
Algorithm ceFace(a, b)
        If  b = 0  then
                Return 1
        EndIf
        c ← ceFace(a, b DIV 2)
        If   b MOD 2 = 0   then
                Return c * c
        Else
               Return  c * ceFace(a, b DIV 2) * a
        EndIf
EndAlgorithm
```

Precizati care dintre urmatoarele afirmatii sunt adevarate referitor la apelul algoritmului *ceFace(a, b)*.

A. Pentru apelul *ceFace(2, 14)* algoritmul returneaza 16384, algoritmul autoapelandu-se de 14 ori.

B. Algoritmul returneaza a^b.

Raspuns: A fals (functia ceFace(2,14) returneaza 2^{14} = 16384 dar se autoapeleaza de 15 ori), B adevarat.

Apeluri functie ceFace() si tipariri in ordine temporala (ordinea temporala este ordinea verticala):

```
ceFace(2,14)
c = ceFace(2, 7)     return c*c = 128 * 128
        c = ceFace(2, 3) = 8
                c = ceFace(2,1) = 2
                        c = ceFace(2,0) = 1
                        return ceFace(2, 0) * ceFace(2, 0) * 2 = 1 * 1 * 2
                return ceFace(2,1) * ceFace(2, 1) * 2 = 2 * 2 * 2 = 8
                        c = ceFace(2,1)
                                c = ceFace(2,0) = 1
                                return ceFace(2, 0) * ceFace(2, 0) * 2 = 1 * 1 * 2
        return ceFace(2, 3) * ceFace(2, 3) * 2 = 8 * 8 * 2 = 128      <- 2^7
                c = ceFace(2,1) = 2
                        c = ceFace(2,0) = 1
                        return ceFace(2, 0) * ceFace(2, 0) * 2 = 1 * 1 * 2
                return ceFace(2, 1) * ceFace(2, 1) * 2 = 2 * 2 * 2 = 8
                        c = ceFace(2,1)
                                c = ceFace(2,0) = 1
                                return ceFace(2, 0) * ceFace(2, 0) * 2 = 1 * 1 * 2
```

10. Se considera algoritmul ***ceFace(n)***, unde ***n*** este un numar natural nenul ($1 \leq n \leq 106$).

```
Algorithm ceFace(n)
    k ← 0
    For  i ← 1, n   execute
        p ← i
        While  p > 0  execute
                k ← k + p MOD 2
                p ← p DIV 2
        EndWhile
    EndFor
    Return k
EndAlgorithm
```

Precizati ce se afiseaza in urma apelului ceFace(2046).

A. 11242

B. 11257

C. 11249
D. 11253

Raspuns:
Algoritmul calculeaza suma tuturor valorilor de 1 din reprezentarea in baza 2 a numerelor 1,2,…,n.
Exemplu: pentru n = 8 = 2^3 avem:

numar	reprezentare binara
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Grupam reprezentarile dupa formatul: primul cu ultimul, al doilea cu penultimul, etc.
0 si 7 = 000, 111 ⇒ 3 biti de 1
1 si 6 = 001, 110 ⇒ 3 biti de 1
2 si 5 = 010, 101 ⇒ 3 biti de 1
3 si 4 = 011, 100 ⇒ 3 biti de 1
Observam ca se obtin 4 grupe a cate 3 biti de 1, adica:
4 × 3 = 12 biti de 1 pentru toate numerele naturale pana la 7. Mai trebuie sa adaugam numarul de biti din reprezentarea lui 8, adica 1 bit ($8 = 1000_{(2)}$).
⇒ 12 + 1 = 13 biti de 1
Prin urmare ceFace(8) returneaza valoarea 13.

Pe caz general, trebuie sa gasim cea mai apropiata putere de 2 de numarul nostru si sa aplicam regula. Pentru n = 2k ⇒ n/2 × k + 1. Pentru n = 2046, cea mai apropiata putere de 2 este 2048 = 2^{11} ⇒ 1024 × 11 + 1 = 11265. Atentie, aceasta este suma pentru toate numerele de la 1 pana la 2048. Trebuie sa scadem numarul de biti 1 din reprezentarile 2047 si 2048, adica:
$2048 = 100000000000_{(2)}$ si $2047 = 11111111111_{(2)}$ ⇒ 11265 − 1 − 11 = 11253. Acesta este si raspunsul final: 11253 (varianta D).

11. Se considera algoritmul ***F(n, k)***, unde ***n*** si ***k*** sunt numere naturale ($1 \leq n \leq 10^9$, $0 \leq k \leq n$).

```
Algorithm F(n, k)
    If  k > (n DIV 2)   then
            k ← n - k
    EndIf
    r ← 1
    For  i ← 0, k - 1   execute
            r ← r  *  (n − i)  DIV  (i + 1)
```

```
    EndFor
    Return r
EndAlgorithm
```

Care dintre urmatoarele afirmatii sunt adevarate?
A. Algoritmul calculeaza numarul submultimilor ordonate cu **k** elemente dintr-o multime cu **n** elemente
B. Algoritmul efectueaza 3 inmultiri pentru apelul **ceFace(100, 97)**
C. Algoritmul optimizeaza calculul coeficientului binomial aplicand proprietatea de simetrie **C(n, k) = C(n, n − k)**
D. Algoritmul efectueaza 97 inmultiri pentru apelul **ceFace(100, 97)**

Raspuns: B si C.
Algoritmul implementat calculeaza coeficientul binomial folosind relatia combinatoriala $C_n^k = \frac{n!}{k!(n-k)!}$. Algoritmul ajusteaza k la 100 − 97 = 3 deoarece 97 > 50. Bucla ruleaza 3 iteratii, fiecare implicand o singura inmultire. Algoritmul foloseste relatia C(n, k) = C(n, n − k) pentru a efectua mai putine calcule.