Simulare Examen Admitere UBB 2024 – 10 Februarie 2024
Proba scrisa la Informatica

NOTA IMPORTANTA:
In lipsa altor precizări:

- Presupuneți ca toate operațiile aritmetice se efectuează pe tipuri de date nelimitate (nu exista overflow / underflow)
- Numerotarea indicilor tuturor șirurilor / vectorilor începe de la 1
- Toate restricțiile se refera la valorile parametrilor actuali la momentul apelului inițial.

1. Se considera subalgoritmii **F1** si **F2** care primesc ca parametrii 2 numere naturale **a** si **b** (1 ≤ **a** ≤ 1000, 1 ≤ **b** ≤ 1000).

```
Subalgorithm F1(a, b):
    If b = 0 then
        return 1
    EndIf
    If b MOD 2 = 0 then
        return F1(a, b/2) * F1(a, b/2)
    Else
        return F1(a, b/2) * F1(a, b/2) * a
    EndIf
EndSubalgorithm

Subalgorithm F2(a, b):
    If b = 0 then
        return 1
    EndIf
    put ← F2(a, b / 2)
    If b MOD 2 = 0 then
        return put * put
    Else
        return put * put * a
    EndIf
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt corecte referitoare la cele 2 subprograme F1 si F2:

a) Subprogramele F1 si F2 au aceiași complexitate timp.
b) Subprogramul F1 se încadrează in clasa de complexitate O(b).
c) Subprogramul F2 se încadrează in clasa de complexitate O($\log_2$b).
d) Cele 2 subprograme calculează valori diferite.

2. Se da următoarea expresie logica:

(A **AND** B) **OR** (**NOT** B **AND** C) **OR** ((A **AND NOT** C) **AND NOT** B)

Precizați pentru ce valori ale lui A, B si C, expresia are valoare de adevăr TRUE:

a) A – TRUE, B – TRUE, C – FALSE
b) A – TRUE, B – FALSE, C – FALSE
c) A – FALSE, B – FALSE, C – FALSE
d) A – TRUE, B – FALSE, C – TRUE

3. Precizați care dintre următoarele expresii logice verifica corect daca variabila N (număr natural) are ultima cifra 0 sau 5:

a) (N **MOD** 6 = 0 **AND** N **MOD** 10 ≠ 6) **OR** (N **MOD** 5 = 0 **AND** N **MOD** 2 = 1)
b) (N **MOD** 10 = 0 **OR** N **MOD** 15 = 0)
c) ((N **MOD** 10) **MOD** 3 = 2 **AND** N **MOD** 4 = 1) **OR** (N **MOD** 2 = 0 **AND** N **MOD** 5 = 0)
d) (N **MOD** 10 = 0 **OR** N **MOD** 10 = 5)

4. Subalgoritmul **suma** primește ca parametru de intrare un număr natural **n** (1 ≤ **n** ≤ 1000).

```
Subalgorithm suma(n):
    If n = 1 then
        return 1
    EndIf
    return n + n * suma(n - 1)
EndSubalgorithm
```

Precizați ce va returna subalgoritmul in funcție de valoarea inițiala a parametrului **n**:

a) n!
b) n + n!
c) n! / 0! + n!/1! + … + n! / (n-1)!
d) $\sum_{k=1}^{n} \frac{n!}{(k-1)!}$

5. Se considera un sir v cu n (1 ≤ n ≤ 1000) elemente numere naturale. Pentru fiecare element din sir, se dorește identificarea următorului element mai mare decât el, existent in sir. Precizați care informații sunt **false**:

a) Este imposibila rezolvarea acestei probleme într-o complexitate liniara.
b) O varianta de rezolvare ar fi sa inițiem o parcurgere spre dreapta începând din fiecare element al șirului in încercarea de a identifica următorul element mai mare.
c) Cea mai buna soluție existenta implica o complexitate pătratica.
d) Cea mai buna soluție implica o complexitate liniara.

6. Se considera subalgoritmul *scrie* care primește ca parametru unic de intrare un număr natural n (1 ≤ **n** ≤ 1000).

```
Subalgorithm scrie(n):
    If n > 0 then
        scrie(n-1)
        write n, ' '
        scrie(n-1)
    EndIf
EndSubalgorithm
```

Precizați care dintre următoarele variante de răspuns sunt adevărate:

a) Pentru n = 3, se afișează 1 2 1 3 1 2 1
b) Pentru n = 0, se afișează 0
c) Pentru n = 2, se afișează 2 1 1 2
d) Oricare ar fi n, se afișează 1 2 1 n 1 2 1

7. In urma întârzierii la un meci de fotbal, constați ca scorul este n – m . Desigur, in calitate de microbist, te întrebi in cate moduri s-ar fi putut ajunge la acest scor. Prin numărul de moduri se înțelege numărul de succesiuni diferite de goluri marcate de cele 2 echipe pentru a se ajunge la scorul respectiv. Spre exemplu: 2-2 s-ar fi putut obține in 6 succesiuni diferite.
Precizați care dintre următoarele afirmații sunt adevărate:

a) Scorul 5 – 5 s-ar fi putut obține prin 250 de moduri.
b) Scorul 3 – 4 s-ar fi putut obține prin 35 de moduri.
c) Scorul 4 – 5 s-ar fi putut obține prin 126 de moduri.
d) Scorul 4 – 4 s-ar fi putut obține prin 68 de moduri.

8. Precizați care dintre următoarele secvențe de cod calculează corect numărul de cifre ale unui număr natural transmis ca parametru (0 ≤ n ≤ 1.000.000.000):

a)
```
Subalgorithm nrCif(n):
    If n = 0 then
        return 0
    EndIf
    return 1 + nrCif(n DIV 10)
EndSubalgorithm
```

b)
```
Subalgorithm nrCif(n):
    If n = 0 then
        return 1
    EndIf
    return 1 + nrCif(n DIV 10)
EndSubalgorithm
```

c)
```
Subalgorithm nrCif(n):
    cnt ← 0
    While n ≠ 0 execute
        cnt ← cnt + 1
        n ← n DIV 10
    EndWhile
    return cnt
EndSubalgorithm
```

d)
```
Subalgorithm nrCif(n):
    cnt ← 0
    Execute
        cnt ← cnt + 1
        n ← n DIV 10
    While n ≠ 0
    return cnt
EndSubalgorithm
```

9. Se considera un tip de date natural care se scrie pe **x** biți. Acesta poate lua valori din intervalul:

a) $[-2^x, -2^{x-1}-1]$
b) $[0, 2^{x-1}-1]$
c) $[0, 2^x-1]$
d) $[-2^{x-1}, 2^{x-1}-1]$

10. Se considera subprogramul ***f*** care primește ca parametru unic de intrare variabila **n** ($1 \le$ **n** ≤ 1000).

```
Subalgorithm f(n):
    For i ← 1, n execute
        k ← i
        For j ← 1, k execute
            k ← k DIV 2
        EndFor
    EndFor
EndSubalgorithm
```

Precizați in ce clasa de complexități se încadrează algoritmul de mai sus:

a) $O(n * \log_2(n))$
b) $O(\log_2 n)$
c) $O(\log_2(n!))$
d) $O(n^2)$

11. Subalgoritmul *fct* primește ca parametru unic de intrare un număr natural nenul **n** ($1 \leq$ **n** ≤ 10.000).

```
Subalgorithm fct(n):
    p ← 1
    While n > 0 execute
        If n MOD 2 = 1 then
            Write p, ' '
        EndIf
        p ← p * 2
        n ← n DIV 2
    EndWhile
EndSubalgorithm
```

Care este efectul produs de subalgoritmul *fct*?

a) Îl scrie pe n in baza 2 si afișează pozițiile biților egali cu 1.
b) Determina si afișează termenii sumei puterilor lui 2 care au ca rezultat valoarea n.
c) Îl împarte pe n la 2 pana când n-ul ajunge egal cu 0 si la fiecare pas, daca n este impar, afișează 2^{pas-1}.
d) Se afișează scrierea lui n in baza 2.

12. Pentru afișarea unei clepsidre de dimensiune n, se utilizează caractere ***** si **#**. Modul de generare este următorul (pentru n = 5 se afișează):

```
*********
#*******#
##*****##
###***###
####*####
###***###
##*****##
#*******#
*********
```

Utilizând acest mod de generare, precizați care dintre următoarele afirmații sunt adevărate:

a) Pentru n = 7, se utilizează 98 de "*" si 72 de "#"
b) Pentru n = 10, se utilizează 199 de "*" si 162 de "#"
c) Pentru n = 1, se utilizează o "*" si un "#"
d) Pentru n = 7, se utilizează 99 de "*" si 72 de "#"

13. Se considera subalgoritmul *produs* care primește ca si parametrii de intrare o matrice **a** cu **n** linii si **n** coloane si variabila **n**, număr natural nenul (1 ≤ n ≤ 1000).

```
Subalgorithm produs(a, n):
    p ← 1
    i ← 2
    While i < n execute
        j ← 1
        While j < i AND j < n – i + 1 execute
            p ← p * a[i][j]
            j ← j + 1
        EndWhile
        i ← i + 1
    EndWhile
    return p
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate referitor la ce calculează subalgoritmul produs(a, n):

a) Subalgoritmul calculează produsul elementelor situate deasupra diagonalei principale si deasupra diagonalei secundare.
b) Subalgoritmul calculează produsul elementelor situate sub diagonala principala si deasupra de diagonala secundara.
c) Subalgoritmul calculează produsul elementelor situate deasupra de diagonala principala si sub diagonala secundara.
d) Nicio afirmație nu descrie corect ce calculează subprogramul.

14. Se considera următorul subprogram *f* care primește ca parametru unic un număr natural nenul **n** (1 ≤ **n** ≤ 1.000.000).

```
SubAlgorithm f(n):
    F[] ← {0,0,0,0,0,0,0,0,0,0,0}
    nr_c ← 0 ; m ← 0
    While n ≠ 0 execute
        F[n MOD 10] ← 1
        n ← n DIV 10
    EndWhile
    For i ← 1, 9 execute
        If F[i] = 0 then
            nr_c ← nr_c + 1
            a[nr_c] ← i
        EndIf
    EndFor
    For i ← nr_c, 1, -1 execute
        m ← m * 10 + a[i]
    EndFor
    return m
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt **corecte**:

a) Secvența alăturata utilizează un algoritm de sortare prin metoda numărării.
b) Subalgoritmul calculează cel mai mare număr care se poate obține din cifrele care apar in n.
c) Subalgoritmul calculează cel mai mare număr care se poate obține din cifrele distincte care nu apar in n.
d) Subalgoritmul calculează cel mai mare număr care se poate obține utilizând toate cifrele distincte nenule care nu apar in **n**.

15. Se considera o matrice de n*m elemente numere naturale. Se dorește stocarea acestei matrici ca vector. Pentru a se face acest lucru, vom folosi un sistem de asociere a valorilor din matrice, cu valorile din vectorul asociat. Ex. A[1][1] => V[1], A[1][m] => V[m], A[2][2] => V[m + 2], … . Dandu-se aceasta metoda de tranziție, trebuie pentru un element din matrice sa deduceți cei 4 vecini ai săi, sus, jos, stânga, dreapta, ca si coordonate pe vectorul V. De exemplu, daca vi se da A[i][j], trebuie sa deduceți pozițiile in vectorul V a elementelor A[i-1][j], A[i+1][j], A[i][j-1], A[i][j+1].

Precizați care dintre următoarele afirmații sunt corecte:

a) A[i][j] => VECIN SUS = V[(i-2)*m + j], VECIN JOS = V[i * m + j], VECIN STANGA = V[(i-1)*m + j - 1], VECIN DREAPTA = V[(i-1)*m + j].
b) A[i][j] => VECIN SUS = V[(i-1)*m + j], VECIN JOS = V[(i+1) * m + j], VECIN STANGA = V[i*m + j - 1], VECIN DREAPTA – V[i*m + j + 1].
c) A[i][j] => VECIN SUS = V[(i-2)*m + j], VECIN JOS = V[i * m + j], VECIN STANGA = V[(i-1)*m + j - 1], VECIN DREAPTA = V[(i-1)*m + j + 1].
d) A[i][j] => VECIN SUS = V[(i-1)*m + j], VECIN JOS = V[(i+1) * m + j], VECIN STANGA = V[i*m + j - 1], VECIN DREAPTA = V[i*m + j].

16. Se considera cele 2 subprograme de mai jos ceFace1 si ceFace2 care primesc ca parametru unic de intrare valoarea întreaga n (1 ≤ n ≤ 1.000.000).

```
Subalgorithm ceFace1(n):
    s ← 1; d ← n
    While s ≤ d execute
        m ← (s + d) / 2
        If m * m * m = n then
            return 1
        EndIf
        If m * m * m > n then
            d ← m – 1
        Else
            s ← m + 1
        EndIf
    EndWhile
    return 0
EndSubalgorithm
```

```
Subalgorithm ceFace2(n):
    i ← 1
    While i * i * i < n execute
        i ← i + 1
    EndIf
    If i * i * i = n then
        return 1
    EndIf
    return 0
EndSubalgorithm
```

Cele 2 secvențe de cod:

a) Returnează radical de ordin 3 din n.
b) Verifica daca n este cub perfect.
c) Utilizează 2 algoritmi de complexități diferite.
d) Au complexități similare.

17. Precizați care dintre următorii subalgoritmi calculează corect C_n^k:
(Acolo unde este prezenta, matricea **Comb** se considera a avea toate elementele nule înainte de apelul inițial si un număr infinit de elemente. **n** si **k** la apelul inițial au valori valide pentru a se putea calcula C_n^k)

a)
```
Subalgorithm C(n, k):
    If n = 1 OR n = k OR k = 0 then
        return 1
    EndIf
    If Comb[n-1][k-1] = 0 then
        Comb[n-1][k-1] ← C(n-1, k-1)
    EndIf
    If Comb[n-1][k] = 0 then
        Comb[n-1][k] ← C(n-1, k)
    EndIf
    return Comb[n-1][k-1] + Comb[n-1][k]
EndSubalgorithm
```

b)
```
Subalgorithm C(n, k):
    prod ← 1
    For i ← n – k + 1, n execute
        prod ← prod * i
    EndFor
    For i ← 1, k execute
        prod ← prod DIV i
    EndFor
    return prod
EndSubalgorithm
```

c)
```
Subalgorithm C(n, k):
    If n = 0 OR k = 0 then
        return 1
    EndIf
    return C(n – 1, k - 1) + C(n – 1, k)
EndSubalgorithm
```

d)
```
Subalgorithm C(n, k):
    b ← 1
    For i ← 1, k execute
        b ← b * i
    EndFor
    a ← b
    For i ← k + 1, n execute
        a ← a * i
    EndFor
    return a DIV b
EndSubalgorithm
```

18. Se considera 3 subalgoritmi *ceFace1*, *ceFace2* si *ceFace3* care primesc ca parametrii un vector de numere naturale nenule a (1 ≤ a[i] ≤ 1.000.000 ∀ i, 1 ≤ i ≤ n) si 3 numere naturale nenule n, i si j (2 ≤ n ≤ 1000).

```
Subalgorithm ceFace1(a, n, i, j):
    If i < n then
        If j ≤ n then
            If a[i] > a[j] then
                a[i] ↔ a[j]
            EndIf
            ceFace1(a, n, i, j + 1)
        Else
            ceFace1(a, n, i + 1, i + 1)
        EndIf
    EndIf
EndSubalgorithm
```

```
Subalgorithm ceFace2(a, n, i, j):
    If i < n then
        If j ≤ n then
            If a[j] < a[j-1] then
                a[i] ↔ a[j]
            EndIf
            ceFace2(a, n, i, j+1)
        Else
            ceFace2(a, n, i+1, 2)
        EndIf
    EndIf
EndSubalgorithm
```

```
Subalgorithm ceFace3(a, n, i, j):
    If i < n then
        If j ≥ 2 then
            If a[j] < a[j-1] then
                a[j] ↔ a[j-1]
            EndIf
            ceFace3(a, n, i, j-1)
        Else
            ceFace3(a, n, i+1, i+1)
        EndIf
    EndIf
EndSubalgorithm
```

Considerând mereu apelurile inițiale de forma *ceFace1*(a, n, 1, 1), *ceFace2*(a, n, 1, 2), *ceFace3*(a, n, 1, 1) precizați care dintre următoarele afirmații sunt **adevărate**:

a) Subalgoritmul ceFace1 sortează vectorul **a** descrescător folosind metoda de sortare Selection Sort.

b) Subalgoritmii ceFace1 si ceFace2 sortează crescător elementele vectorului a si utilizează aceiași metoda de sortare.

c) Subalgoritmul ceFace1 sortează folosind metoda de sortare Selection Sort, iar subalgoritmul ceFace3 sortează utilizând metoda de sortare Insertion Sort.

d) Toți algoritmii sortează crescător elementele șirului **a** utilizând algoritmi de sortare diferiți.

19. Se considera subalgoritmii *f1* si *f2* care primesc ca parametru unic de intrare numărul natural nenul **n** (100 ≤ **n** ≤ 100.000.000).

```
Subalgorithm f1(n):
    If n = 0 then
        return 0
    EndIf
    If n MOD 2 = 0 then
        return f1(n DIV 10) * 10 + n MOD 10
    EndIf
    return f1(n DIV 10)
EndSubalgorithm

Subalgorithm f2(n):
    If n = 0 then
        return 0
    EndIf
    If n MOD 2 = 0
        return f2(n DIV 10) * 100 + (n MOD 10) * 10 + n MOD 10
    EndIf
    return f2(n DIV 10) * 10 + n MOD 10
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt **adevărate**:

a) f1(12345) = 124, f2(12345) = 1223445
b) f1(1) = 1, f2(1) = 1
c) f2(1357) = 11335577, f1(12) = 2
d) Algoritmul f1 calculează si returnează valoarea numărului n in urma eliminării tuturor cifrelor sale impare iar algoritmul f2 calculează si returnează valoarea numărului n in urma dublării tuturor cifrelor sale pare.

20. Se considera cele 4 variante de implementare ale unei funcții care calculează numărul de divizori ai unui număr natural nenul **n** ($1 \le$ **n** $\le 1.000.000$).

```
Subalgorithm nrDiv1(n):
    cnt ← 0
    For d ← 1, n execute
        If n MOD d = 0 then
            cnt ← cnt + 1
        EndIf
    EndFor
    return cnt
EndSubalgorithm

Subalgorithm nrDiv2(n):
    cnt ← 0
    For i ← 1, n execute
        If n MOD i = 0 then
            cnt ← cnt + 2
        EndIf
    EndFor
    return cnt DIV 2
EndSubalgorithm

Subalgorithm nrDiv3(n):
    d ← 2
    nrdiv ← 1
    While n > 1 execute
        p ← 0
        While n MOD d = 0 execute
            p ← p + 1
            n ← n DIV d
        EndWhile
        nrdiv ← nrdiv * (p + 1)
        d ← d + 1
        If d * d > n then
            d ← n
        EndIf
    EndWhile
    return nrdiv
EndSubalgorithm
```

```
Subalgorithm nrDiv4(n):
    cnt ← 2
    For d ← 2, √n execute
        If n MOD d = 0 then
            cnt ← cnt + 2
        EndIf
        If d = n DIV d then
            cnt ← cnt – 1
        EndIf
    EndFor
    return cnt
EndSubalgorithm
```

Precizați care dintre cei 4 subalgoritmi calculează corect numărul de divizori ai unui număr care respecta condițiile din enunț:

a) Algoritmii nrDiv1, nrDiv2 si nrDiv3.
b) Algoritmii nrDiv1, nrDiv2 si nrDiv4.
c) Doar algoritmii nrDiv1 si nrDiv2.
d) Toți cei 4 subalgoritmi.

21. Precizați care dintre următoarele afirmații sunt adevărate referitoare la complexitatile celor 4 subalgoritmi prezenți la grila 20.

a) Algoritmii nrDiv1 si nrDiv2 au aceiași complexitate timp.
b) Algoritmii nrDiv3 si nrDiv4 au aceiași complexitate timp.
c) Algoritmul nrDiv3 are complexitate timp mai buna decât toți ceilalți.
d) Toate informațiile sunt corecte.

22. Se consideră un arbore cu rădăcină în care doar 134 dintre nodurile arborelui au exact 2 descendenți direcți (fii), restul nodurilor având cel mult un descendent direct (fiu). Care este numărul frunzelor arborelui?

a) 2^{134}
b) $2^{134} - 1$
c) 134
d) 135

23. Câte noduri are un arbore cu rădăcină în care numărul de muchii este egal cu numărul de muchii ale unui graf neorientat complet cu **n** noduri.

a) $\frac{n*(n+1)}{2} + 1$
b) $\frac{n*(n-1)}{2} + 1$
c) $\frac{n*n-n+2}{2}$
d) Nicio formula nu calculează corect răspunsul.

24. Se considera cei 2 subalgoritmi *afis* care primește ca si parametri de intrare un sir **a** cu **n** elemente numere naturale nenule ($1 \le a[i] \le 1.000.000$, $1 \le$ **n** ≤ 100) si subprogramul *f1* care primește ca si parametri de intrare șirul **a** si încă doua numere naturale nenule.

```
Subalgorithm f1(a, n, cnt):
    If cnt != 0 then
        f1(a, n DIV 2, cnt - 1)
        If n MOD 2 = 1 then
            write a[cnt], ' '
        EndIf
    EndIf
EndSubalgorithm

Subalgorithm afis(a, n):
    For i ← 0, 2^n – 1 execute
        f1(a, i, n)
        write '\n'
    EndFor
EndSubalgorithm
```

Precizați care dintre următoarele informații sunt adevărate in legătura cu subprogramul *afis*(a, n):

a) Subalgoritmul generează si afișează pe ecran toate numerele de la 0 pana la $2^n – 1$ scrise in baza 2.
b) Subalgoritmul generează si afișează pe ecran toate submulțimile mulțimii formate din elementele șirului a.
c) Subalgoritmul intra in bucla infinita.
d) Nicio afirmație nu este corecta

Număr grila	Răspuns corect
Grila 1	B, C
Grila 2	A, B, D
Grila 3	D
Grila 4	C, D
Grila 5	A, C
Grila 6	A
Grila 7	B, C
Grila 8	D
Grila 9	B, C
Grila 10	A, C
Grila 11	B, C
Grila 12	B
Grila 13	B
Grila 14	A, D
Grila 15	C
Grila 16	B, C
Grila 17	A, B
Grila 18	C, D
Grila 19	D
Grila 20	A
Grila 21	A, B
Grila 22	D
Grila 23	B, C
Grila 24	B