Simulare Examen Admitere UBB 2024 – 13-14 Iulie 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 subalgoritmul *ce_face* care primește ca si parametru unic de intrare un număr natural nenul **n** ($1 \le \mathbf{n} \le 10^9$).

```
Subalgorithm ce_face(n):
        a ← 0
        b ← 0
        d1 ← 2
        d2 ← 5
        While d1 ≤ n OR d2 ≤ n  execute
                a ← a + n DIV d1
                b ← b + n DIV d2
                d1 ← d1 * 2
                d2 ← d2 * 5
        EndWhile
        If a < b then
                return a
        Else
                return b
        EndIf
EndSubalgorithm
```

Precizați care dintre următoarele informații referitoare la subalgoritmul *ce_face* sunt adevărate:

A. Pentru n = 10 se returnează valoarea 3.
B. Subalgoritmul determina in cati de 0 se termina numarul n!.
C. Mereu se returnează valoarea lui b.
D. Subalgoritmul calculează puterea maxima **p** pentru care 10^p este divizor al numărului n!.

2. Precizați care dintre următoarele expresii verifica corect daca o variabila n de cel mult 3 cifre, număr natural, este multiplu de 9 sau nu.

A. n **MOD** 9 = 0
B. (n **MOD** 10 + (n **DIV** 10) **MOD** 10 + n **DIV** 100) **MOD** 9 = 0
C. (n **MOD** 10 + (n **MOD** 100) **DIV** 10 + n **DIV** 100) **MOD** 9 = 0
D. (n **MOD** 10 + (n **DIV** 10) **MOD** 10 + n **DIV** 100) = 0

3. Se considera algoritmul *ce_face* care primește ca parametru unic de intrare un număr natural nenul **n** ($1 \le \mathbf{n} \le 10^9$).

```
Subalgorithm ce_face(n):
    a ← 0
    b ← 0
    p ← 1
    nr ← 0
    While n > 0 execute
        d ← n MOD 10
        a ← a * 10 + d
        b ← b + d * p
        p ← p * 10
        If a = b then
            nr ← a
        EndIf
        n ← n DIV 10
    EndWhile
    Return nr
EndSubalgorithm
```

Precizați care dintre următoarele afirmații referitoare la subprogramul *ce_face* sunt adevărate.

A. In cazul in care numarul **n** este palindrom, se returneaza valoarea acestuia.
B. Subalgoritmul nu poate returna valoarea 0 indiferent de valoarea inițială a numărului **n** conform restricțiilor.
C. Subalgoritmul calculeaza si returneaza oglinditul numarului n.
D. Subalgoritmul returneaza lungimea celui mai lung sufix palindrom a numarului **n**.

4. Se considera subalgoritmul *prelucrare* care primește ca si parametrii de intrare un sir **a** cu **n** elemente numere naturale ($1 \le \mathbf{n} \le 1000$, $1 \le \mathbf{a}[i] \le 10^9$, $1 \le i \le \mathbf{n}$).

```
Subalgorithm prelucrare(a, n):
    For i ← 1, n execute
        If a[i] MOD 2 = 0 then
            poz ← i
            For j ← i + 1, n execute
                If a[j] MOD 2 = 1 then
                    poz ← j
                EndIf
            EndFor
            a[i] ← a[i] * a[poz]
            a[poz] ← a[i] DIV a[poz]
            a[i] ← a[i] DIV a[poz]
        EndIf
    EndFor
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate referitoare la subalgoritmul *prelucrare*:

A. Subalgoritmul rearanjeaza elementele sirului in asa fel incat toate elementele impare sa fie inaintea celor pare.
B. Subalgoritmul rearanjeaza elementele sirului in asa fel incat toate elementele pare sa fie inaintea celor impare.
C. Subalgoritmul elimina toate elementele pare din sir.
D. Niciuna dintre afirmatii nu este adevarata.

5. Se considera subprogramul *calcul* definit mai jos care primește ca si parametru de intrare 4 numere naturale nenule **x, y, z, t** (1 ≤ **x, y, z, t** ≤ 1000).

```
Subalgorithm calcul(x, y, z, t):
    If x = 0 then
        Return 1
    EndIf
    If x > y AND y > z AND z > t then
        return x + calcul(x – 1, y + z, z + t, t * 2)
    Else
        return calcul(x – 1, y * 2, z * 2, t * t)
    EndIf
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt false:

A. La apelul calcul(1, 2, 3, 4) algoritmul intra in bucla infinita.
B. In cazul in care la apelul initial x este mai mic sau egal cu y, subprogramul returneaza valoarea 1.
C. Subalgoritmul calculează valoarea 2^**x**.
D. Complexitatea algoritmului este O(x).

6. Se considera subprogramul recursiv *suma* care primește ca si parametru de intrare un număr natural nenul **n** (1 ≤ n ≤ 1000).

```
Subalgorithm suma(n):
    If n > 0 then
        return -1 * n * suma(n - 1)
    Else
        return 1
    EndIf
EndSubalgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Pentru n = 6 se calculeaza valoarea 720
B. Pentru n = 5 se calculeaza valoarea -120
C. Oricare ar fi n, subalgoritmul calculeaza n!
D. Oricare ar fi n, subalgoritmul calculeaza $-1^n * n!$

7. Se considera subprogramul recursiv *suma* care primește ca si parametrii de intrare doua numere naturale nenule **n** si **k** (1 ≤ **n, k** ≤ 1000).

```
Subalgorithm suma(n, k):
    If n = 0 then
        Return 0
    EndIf
    Return n MOD k + suma(n DIV k)
EndSubalgorithm
```

Pentru care dintre următoarele valori ale lui **n** si **k** se afișează ceea ce este precizat?

A. Pentru **n** = 1324 si **k** = 7 se afișează valoarea 10.
B. Pentru **n** = 243 si **k** = 4 se afișează valoarea 9.
C. Pentru **n** = 324 si **k** = 6 se afișează valoarea 3.
D. Pentru **n** = 432 si **k** = 7 se afișează valoarea 12.

8. Se considera subprogramul *f* care primește ca si parametru unic de intrare un număr natural nenul **n** (1 ≤ **n** ≤ 10^9).

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

Precizați care dintre următoarele afirmații sunt adevărate referitoare la secvența de cod de mai sus:

A. Subalgoritmul calculeaza numarul de divizori ai numarului n.
B. Subalgoritmul determina numarul de numere prime cu n din intervalul [1, n].
C. Subalgoritmul determina numarul obtinut din inmultirea divizorilor primi a lui n la puterea 1.
D. Subprogramul determina suma divizorilor numarului n.

9. Se considera subprogramul *f* care are ca parametru unic de intrare un număr natural nenul **n** ($0 \leq \mathbf{n} \leq 10^9$).

```
Subalgorithm f(n):
    cnt ← 0
    For i ← 1, n execute
        ok ← 1
        For j ← 1, i execute
            If i MOD j = 0 then
                ok ← 0
            EndIf
        EndFor
        If ok = 1 then
            Break
        EndIf
    EndFor
EndSubalgorithm
```

In ce clasa de complexitate se incadreaza acest algoritm?

A. O(1)
B. O(n)
C. $O(n^2)$
D. O(n * (n - 1) / 2)

10. Se considera subalgoritmul *exp* care primește ca si parametrii de intrare 2 numere naturale nenul **a** si **b** ($1 \leq a, b \leq 1000$).

```
Subalgorithm exp(a, b):
    p ← b
    rez ← 0
    While a != 0 execute
        If a MOD 2 = 1 then
            rez ← rez + p
        EndIf
        a ← a DIV 2
        p ← p * 2
    EndWhile
    Return rez
EndSubalgorithm
```

Care dintre următoarele afirmații sunt adevărate?

A. Algoritmul calculeaza a^b.
B. Algoritmul calculeaza b * a^b.
C. Algoritmul calculeaza a * b.
D. Algoritmul calculeaza b^a.

11. Se considera subprogramul *verificare* care primeste ca si parametrii de intrare un numar natural nenul **n** si 2 parametrii **s** si **d**. ($1 \leq \mathbf{n, s, d} \leq 10000$)

```
Subalgorithm verificare(n, s, d):
    If s = d then
        return n MOD s != 0
    EndIf
    m ← (s + d) DIV 2
    return verificare(n, s, m) AND verificare(n, m + 1, d)
EndSubalgorithm
```

Precizați care este efectul subalgoritmului *verificare*(n, 2, n - 1):

A. Subalgoritmul verifica daca exista cel putin un numar prim in intervalul [2, n-1].
B. Subalgoritmul verifica daca numarul n este numar prim.
C. Subalgoritmul are o complexitate timp logaritmica.
D. Subalgoritmul verifica daca in intervalul [2, n-1] exista vreun numar par.

12. Se considera subprogramul *uemm*(a, n) care primeste ca si parametrii de intrare un sir **a** cu **n** elemente numere naturale nenule ($1 \le n \le 1000$, $1 \le a[i] \le 10^9$)

```
Subalgoritm uemm(a, n)
    s ← [0] * 1000
    h ← 0
    For i ← 1, n execute
        While h > 0 AND a[s[h]] < a[i] execute
            a[s[h]] ← a[i]
            h ← h – 1
        EndWhile
        s[h + 1] ← i
        h ← h + 1
    EndFor
    While h > 0 execute
        a[s[h]] ← -1
        h ← h - 1
    EndWhile
EndSubalgoritm
```

Care dintre urmatoarele afirmatii sunt adevarate referitoare la secvente de cod de mai sus?

A. Subalgoritmul inlocuieste fiecare element din sirul a cu o valoarea mai mare decat ea.
B. Subalgoritmul determina si inlocuieste fiecare elemente cu urmatorul element mai mare in sir, iar daca un asemenea element nu exista se va inlocui cu valoarea -1.
C. Subalgoritmul are o complexitate patratica.
D. Subalgoritmul determina si inlocuieste fiecare element cu valoarea maxima din sir.

13. Se doreste interclasarea a 10 siruri de numere ordonate crescator. Pentru a se realiza acest lucru, se foloseste o functie de interclasare care pentru oricare 2 siruri transmise ca parametrii determina sirul ordonat care contine elementele celor 2 siruri cu un numar de pasi egal cu n + m unde n si m sunt dimensiunile sirurilor de intrare. Stiind ca dimensiunile celor 10 sirurilor sunt (123, 43, 254, 154, 235, 75, 234, 153, 753, 432), care este numarul minim de pasi pe care trebuie sa-i facem pentru a determina sirul ordonat care sa contina toate elementele acestor siruri?
    A. 2656
    B. 2456
    C. 7281
    D. 9256

14. Se consideră graful neorientat definit prin mulţimea vârfurilor {1,2,3,4,5,6} şi mulţimea muchiilor {[1,2],[2,3],[3,4],[3,5],[4,5],[1,3],[2,6],[2,4],[4,6]}. Care este numărul minim de muchii ce pot fi eliminate astfel încât graful parţial obţinut să nu mai fie conex?
    A. 2
    B. 3
    C. 4
    D. 5

15. Se consideră următorul algoritm *lps*(s, n), unde s este un șir de caractere de lungime n.

```
Algorithm lps(s, n):
    dp ← [0 * (n + 1)] * (n + 1)
    For i = 1 to n execute
        dp[i][i] ← 1
    For cl = 2 to n execute
        For i = 1 to n - cl + 1 execute
            j ← i + cl - 1
            If s[i-1] = s[j-1] AND cl = 2 then
                dp[i][j] ← 2
            ElseIf s[i-1] = s[j-1] then
                dp[i][j] ← dp[i+1][j-1] + 2
            Else
                dp[i][j] ← max(dp[i][j-1], dp[i+1][j])
            EndIf
        EndFor
    EndFor
    Return dp[1][n]
EndAlgorithm
```

Care este valoarea returnata de subalgoritm la apelul lps("BBABCBAAB", 9)?

A. 5 B.6 C.7 D.8

16. Se considera subprogramul genMat(a, n) care primeste ca si parametrii de intrare o matrice a cu n elemente numere naturale nenule. |x| este valoarea absoluta a lui x. ($1 \le n \le 10^3$)

```
Subalgorithm genMat(a, n)
    For i ← 1, n execute
        For j ← 1, n execute
            a[i][j] ← min(|i - j|, |n + 1 – i – j|)
        EndFor
    EndFor
EndSubalgorithm
```

Care este suma tuturor elementelor din matrice considerand ca la apelul initial n = 10?

A. 160
B. 144
C. 256
D. 200

17. Se considera subalgoritmul cautare(a, st, dr, val) care primeste ca si parametrii de intrare un sir a cu n elemente numere naturale ordonate crescator si o valoare val. ($1 \le n, a[i], val \le 10^3$).

```
Subalgorithm cautare(a, st, dr, val)
    If st > dr then
        return a[dr]
    EndIf
    mij ← (st + dr) DIV 2
    If a[mij] ≤ val then
        return cautare(a, mij + 1, dr, val)
    Else
        return cautare(a, st, mij-1, val)
    EndIf
EndSubalgorithm
```

Precizati care dintre urmatoarele afirmatii sunt adevarate referitor la subalgoritmul cautare(a, 1, n, val):

A. Subalgoritmul determina pozitia celui mai mare element mai mic sau egal cu val din sir.
B. Subalgoritmul determina cel mai mare element mai mic sau egal cu val din sir.
C. Subalgoritmul determina o pozitie pe care apare val in sir in cazul in care val apare in sir.
D. In cazul in care val apare de mai multe ori in sir, se va determina ultima sa aparitie.

18. Se considera subalgoritmul reuniune(a, n, b, m, c, p) care primeste ca si parametrii de intrare 2 siruri **a** cu **n** elemente si **b** cu **m** elemente ordonate strict crescator. Sirul **c** cu **p** elemente reprezinta parametrii de iesire. Functia pop(a, n) returneaza sirul **a** in urma eliminarii primului element din sir.

```
Subalgorithm reuniune(a, n, b, m, c, p)
    If n = 0 AND m = 0 then
        return c
    EndIf
    If n = 0 then
        p ← p + 1
        c[p] ← b[1]
        b ← pop(b, m)
        m ← m – 1
        return reuniune(a, n, b, m, c, p)
    EndIf
    If m = 0 then
        p ← p + 1
        c[p] ← a[1]
        a ← pop(a, n)
        n ← n – 1
        return reuniune(a, n, b, m, c, p)
    EndIf
    If a[1] < b[1] then
        p ← p + 1
        c[p] ← a[1]
        a ← pop(a, n)
        n ← n – 1
    Else
        p ← p + 1
        c[p] ← b[1]
        b ← pop(b, m)
        m ← m – 1
    EndIf
    return reuniune(a, n, b, m, c, p)
EndSubalgorithm
```

Precizati care dintre urmatoarele afirmatii sunt adevarate.

A. Subalgoritmul reuniune(a, n, b, m, c, 0) calculeaza si returneaza sirul c care contine reuniunea tuturor elementelor din sirurile a si b, in ordine crescatoare.
B. Subalgoritmul reuniune(a, n, b, m, c, 0) calculeaza si returneaza sirul c care contine reuniunea tuturor elementelor din sirurile a si b, dar elementele sirului c nu sunt in ordine crescatoare.
C. Subalgoritmul are o complexitate liniara.
D. Subalgoritmul nu obtine reuniunea celor 2 siruri.

19. Se considera urmatorul sir s = 0, 1, 0, 2, 1, 0, 3, 2, 1, 0, 4, 3, 2, 1, 0 ...
Obtinut prin alimitarea sirurilor primelor i numere naturale luate in ordine inversa. Precizati care dintre urmatoarele afirmatii sunt adevarate stiind ca sirul este indexat de la 1.
    A. Al 324-lea termen al sirului este 1.
    B. Al 254-lea termen al sirului este 22.
    C. Al 1000-lea termen al sirului este 15.
    D. Al 245-lea termen al sirului este 8.

20. Se consideră subalgoritmul ***Ep(n, adjMatrix, s, t)*** care primește ca și parametrii de intrare un număr ***n*** egal cu numărul de noduri ale unui graf neorientat reprezentat de matricea ***adjMatrix*** ($10 \leq n \leq 101$) și două numere naturale ***s***,***t*** ($0 \leq s, t \leq n - 1$) Știind că indexarea nodurilor grafului începe de la valoarea 0, și că există cel puțin un nod cu gradul 2, precizați care dintre următoarele afirmații (A,B,C,D) sunt adevărate:
    1) Subalgoritmul va returna mereu false.
    2) Dacă subalgoritmul Ep returnează true, atunci există un drum între s și t.
    3) Subalgoritmul Ep folosește o căutare în adâncime (DFS) pentru a determina existența unui drum între s și t.
    4) Subalgoritmul Ep va returna false dacă nu există niciun drum între s și t.

```
Subalgorithm Ep(n, adjMatrix, s, t)
    visited ←[0] * 101

    Function DFS(node):
        If node = t then
            Return true
        EndIf
        If visited[node] = 1 then
            Return false
        EndIf
        visited[node] ← 1
        For i ← 1, n execute
            If adjMatrix[node][i] = 1 then
                If DFS(i) = true then
                    Return true
                EndIf
            EndIf
        EndFor
        Return false
    Return DFS(s)
EndSubalgorithm
```

A. Doar afirmațiile 1 și 2 respectă cerința.
B. Afirmațiile 2 și 3 respectă cerința.
C. Doar afirmațiile 2, 3 și 4 respectă cerința.
D. Niciuna dintre variantele de mai sus nu e corectă.

21. Se consideră 3 numere în diferite baze de numerație: x = 429 în baza 10, y = 1AD în baza 16 și z = 110101100 în baza 2. Care dintre următoarele afirmații sunt adevărate? (R. Cotoi)

A. x = y și y ≠ z.
B. x = y = z.
C. x ≠ y și y ≠ z.
D. x ≠ y și y = z.

22. Se consideră subalgoritmul **ceFace(n, k)** definit alăturat, care primește ca parametrii două numere naturale **n** și **k**, cel mult **10^9**. Șirul ***a*** este citit de la tastatură în funcția main, iar șirul ***b*** este inițializat cu 0. Stabiliți care este rezultatul subalgoritmului. (R. Cotoi)

```
Algorithm ceFace(n, k):
  For i from 1, n do
    b[i] = b[i - 1] + a[i]
  EndFor
  st = 0
  dr = 0
  c = 0
  For i from k, n do
    If b[i] - b[i - k] > c then
      c = b[i] - b[i - k]
      st = i - k + 1
      dr = i
    EndIf
  EndFor
  For i from st, dr do
    Output a[i], " "
  EndFor
EndAlgorithm
```

A. Subalgoritmul afișează secvența de lungime maximă din șirul ***a*** aflată în ordine crescătoare.
B. Subalgoritmul afișează secvența de lungime și sumă maximă din șirul **a.**
C. Subalgoritmul afișează secvența de lungime ***k*** din șirul ***a*** cu suma elementelor maximă.
D. Subalgoritmul afișează secvența de lungime ***k*** din șirul **a** cu suma elementelor exact egală cu ***k*** (dacă există, altfel **st > dr**, deci nu se va afișa nimic).

23. Se consideră subalgoritmul ceFace(), definit alăturat. Șirul ***a*** de lungime ***n*** este citit de la tastatură, iar șirul ***sp*** este inițializat cu 0. Funcția max(a, b) returnează maximul dintre numerele a și b. (R. Cotoi)

```
Algorithm ceFace():
  c = sp[0] = a[0]
  If c < 0 then
    c = 0
  EndIf
  For i from 1, n-1 do
    c = c + a[i]
    sp[i] = max(sp[i-1], c)
    If c < 0 then
      c = 0
    EndIf
  EndFor
  c = b = a[n-1]
  If c < 0 then
    c = 0
  EndIf
// continuarea se află în dreapta
```

```
d = b + sp[n-2]
  For i from n-2, 1, -1 do
    c = c + a[i]
    If b < c then
      b = c
    EndIf
    If c < 0 then
      c = 0
    EndIf
    d = max(d, b + sp[i-1])
  EndFor
  Output d
EndAlgorithm
```

Care dintre următoarele afirmații sunt adevărate?
A. Subalgoritmul ceFace calculează suma maximă posibilă care se poate obține din două secvențe oarecare din șirul *a*.
B. Subalgoritmul ceFace calculează suma maximă posibilă care se poate obține din două secvențe care nu au niciun element comun din șirul *a*.
C. Subalgoritmul ceFace calculează secvența de lungime maximă și sumă maximă din șirul *a*.
D. Subalgoritmul ceFace calculează numărul de secvențe cu suma și lungimea maximă din șirul *a*.

24. Se consideră un șir de **n** numere naturale ***a***. Elementele din **a** sunt ordonate crescător folosind metoda **Quick Sort**. Pentru **k** numere **q** introduse de la tastatură, ne dorim să știm numărul maxim de elemente din **a** ale căror sumă nu depășește numărul **q** citit. Care este complexitatea celui mai eficient subprogram care rezolvă problema (k < n)? (R. Cotoi)

A. O(log n)
B. O(n log n)
C. O (n log^2 n)

D. O(n^2 log n)

Numar Grila	Barem
1	BCD
2	ABC
3	A
4	D
5	AC
6	ABD
7	BD
8	B
9	CD
10	C
11	B
12	B
13	C
14	A
15	C
16	A
17	B
18	AC
19	ABD
20	BC
21	A
22	C
23	B
24	B