## 0. Expresii logice

**0.1.** Se consideră expresia logică: **(X OR NOT Z) AND (NOT X OR Y).** Care din variantele de mai jos fac expresia true (adevărată)? (sursa: concurs/admitere la FMI, UBB Cluj-Napoca)

```
A. X ← false;     Y ←false; Z ← true;
B. X ← true;      Y ←false; Z ← false;
C. X ← false;     Y ←true;  Z ← false;
D. X ← true;      Y ←true;  Z ← true.
```

| V(sau/Or) | f | t |
|---|---|---|
| f | f | t |
| t | t | t |

| Λ(şi/and) | f | t |
|---|---|---|
| f | f | f |
| t | f | t |

**0.2.** Precizați care dintre următoarele expresii are valoarea adevărat dacă și numai dacă numărul natural **n** este divizibil cu 3 și are ultima cifră 4 sau 6. Atenţie: **a DIV b = [a/b]** (împărţirea întreagă, b≠0), iar **a MOD b** este restul împărţirii lui **a** la **b,** . (sursa: concurs/admitere la FMI, UBB Cluj-Napoca)

```
A. n DIV 3 = 0 AND (n MOD 10 = 4 OR n MOD 10 = 6);
B. n MOD 3 = 0 AND (n MOD 10 = 4 OR n MOD 10 = 6);
C. (n MOD 3 = 0 AND n MOD 10 = 4) OR (n MOD 3 = 0 AND n MOD 10 = 6);
D. (n MOD 3 = 0 AND n MOD 10 = 4) OR n MOD 10 = 6.
```

**0.3.** Se consideră expresia următoare, în care **a** este un număr natural.

```
((a < 10) AND (a < 5))   OR  ((a > 2)  AND (a ≤ 7))
```

Pentru ce valori ale lui **a** va avea expresia valoarea ***TRUE*** ?

```
A. a ∈ {3,4,5,6,7,8};
B. a ∈ {4,5,6,7};
C. a ∈ {2,3,4,5,6,7};
D. a > 10.
```

**0.4.**Se consideră expresia următoare, în care **a** este un număr natural.

```
((a < 10) OR (a < 5))   AND  ((a > 2)  OR (a ≤ 7))
```

Pentru ce valori ale lui **a** va avea expresia valoarea ***TRUE*** ?

```
A. a ∈ {9,10};
B. a ∈ {11,12,13,14,15};
C. a ∈ {2,3,4};
D. a ∈ {0,1};
```

**0.5.** Se consideră expresia următoare, în care **a** este un număr natural.

**((a < 20) AND (a < 15)) AND ((a > 4) AND (a ≤ 10))**

Pentru ce valori ale lui **a** va avea expresia valoarea ***TRUE*** ?

**A. a ∈ {9,10};**
**B. a ∈ {11,12,13,14,15};**
**C. a ∈ {5,6,7,8,9,10};**
**D. a ∈ {8,9,10,11};**

**0.6.** Se consideră expresia următoare, în care **a** este un număr natural.

**((a < 20) OR (a < 15)) OR ((a < 4) OR (a ≤ 10))**

Pentru ce valori ale lui **a** va avea expresia valoarea ***TRUE*** ?

**A. a ∈ {9,10};**
**B. a ∈ {20,21};**
**C. a ∈ {1,6,17,18,20};**
**D. a ∈ {1,3,5,7,9,11,13,15,17,19};**

**0.7.** Se consideră expresia **(b – a + 1) DIV p.** Cele **3** variabile sunt naturale nenule cu **a<b, p ≠ 0.**

**Care afirmaţii sunt adevărate ?**

- **A.** Pentru **a=1, b=100, p=2,** expresia are valoarea **49**
- **B.** Pentru **a=1, b=100, p=2,** expresia are valoarea **50**
- **C.** Pentru **a=5, b=100, p=6,** valoarea expresiei este egală cu numărul de numere multiplu de **6** dintre **a** şi **b** (inclusiv **a** şi **b**)**;**
- **D.** Pentru **a=6, b= 96, p=6,** valoarea expresiei este egală cu numărul de numere multiplu de **6** dintre **a** şi **b** (inclusiv **a** şi **b**)**.**

**0.8.** Se consideră expresia $C_n^0 + C_n^1 + ... + C_n^n$ , n>0**.**

**Care afirmaţii sunt adevărate ?**

- **A.** Expresia are valoarea $2^{n+1}$ ;
- **B.** Expresia reprezintă numărul de submulţimi ale unei mulţimi de **n** elemente;
- **C.** Expresia are valoarea $2^n$;
- **D.** Expresia este suma coeficienţilor binomiali ai dezvoltării $(a + b)^n$**.**

**0.9.** Se consideră expresia $C_n^0 - C_n^1 + ... + (-1)^n C_n^n$ , n natural nenul**.**

**Care afirmaţii sunt adevărate ?**

- **A.** Expresia are valoarea $2^{n-1}$ ;
- **B.** Expresia reprezintă numărul de submulţimi ale unei mulţimi de n elemente;
- **C.** Expresia are valoarea 0;
- **D.** Expresia este suma coeficienţilor binomiali ai dezvoltării $(a - b)^n$**.**

**0.10.** Se consideră expresia $\mathbf{m^n}$**,** ambele variabile naturale nenule.

**Care afirmaţii sunt adevărate ?**

- **A.** Expresia are valoarea numărului de funcţii injective f: A→ B, cu cardinal(A)=n şi cardinal(B) =m;
- **B.** Expresia are valoarea numărului de funcţii surjective f: A→ B, cu cardinal(A)=n şi cardinal(B) =m;
- **C.** Expresia are valoarea numărului de funcţii f: A→ B, cu cardinal(A)=n şi cardinal(B) =m;
- **D.** Numărul de submulţimi de câte **m** elemente ale unei mulţimi de **n** elemente **(m<n);**

# 1. Cifre/Numere/Diviziblitate/etc.

Atenţie: **a DIV b = [a/b]** (împărţirea întreagă), iar **a MOD b** este restul împărţirii lui **a** la **b**.

**1.1.** Se consideră secvenţa Pseudocod. Ce valoare este afişată? (sursă: personală)

```
            {n,i sunt întregi}
n← 100
i← 0
While i < n execute
      i← i+1
EndWhile
Write i
```

**A.** a lui **n;**
**B.** **99;**
**C.** **101;**
**D.** **100.**

**1.2.** Ce valoare se afişează după execuţia secvenţei? (Pseudocod; sursă: personală)

```
                  {n,i sunt întregi}
n← 50
For  i← 3,n,5 execute
            {corpul ciclului,  nu se modifică i}
EndFor
Write i
```

**A.** **n+3;**
**B.** **50;**
**C.** **53;**
**D.** **n-2**.

**1.3.** Care secvenţă schimbă valorile întregi, nenule, ale lui **a** şi **b**, între ele? (Pseudocod; sursă: personală)

```
A.          {a,b sunt întregi}
   a← a-b
   b← b+a
   a← b-a
```

```
B.          {a,b sunt întregi}
   a← a+b
   b← a-b
   a← a-b
```

```
C.          {a,b sunt întregi}
      a← a DIV b
      b← a*b
      a← b DIV a
```

```
D.          {a,b sunt întregi}
      a← a*b
      b← a DIV b
      a← a DIV b
```

**1.4.** Care afirmaţii sunt adevărate, pentru **n** întreg? (**v** este vector cu **10** elemente). (Pseudocod; sursă:personală)

```
For i←0,9 execute
      v[i]← 0
EndFor
While n > 0 execute
  v[n MOD 10]←1
  n← n DIV 10
EndWhile
```

**A.** Pentru **n = 129912** ⇒ **v= [0, 2, 2, 0, 0, 0, 0, 0, 0, 9];**
**B.** Pentru **n = 13913** ⇒ **v= [0, 1, 1, 0, 0, 0, 0, 0, 0, 1];**
**C.** Pentru **n = -1292121** ⇒ **v= [0, 1, 1, 0, 0, 0, 0, 0, 0, 1];**
**D.** Pentru **n = -17972,** ⇒ **v= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0].**

**1.5.** Care afirmaţii sunt adevărate, pentru **n** întreg? (**v** este vector cu **10** elemente). (Pseudocod; sursă: personală)

```
For i←0,9 execute
      v[i]← 0
EndFor
While n > 0 execute
  v[n MOD 10] ← v[n MOD 10]+1
  n           ← n DIV 10
EndWhile
```

**A.** Pentru **n = 179712**, vectorul **v** are valorile **[0, 1, 1, 0, 0, 0, 0, 1, 0, 1];**
**B.** Pentru **n = 12912**, vectorul **v** are valorile **[0, 1, 1, 0, 0, 0, 0, 0, 0, 1];**
**C.** Pentru **n = 132712**, vectorul **v** are valorile **[0, 2, 2, 1, 0, 0, 0, 1, 0, 0];**
**D.** Pentru **n = -132912**, vectorul **v** are valorile **[0, 0, 0, 0, 0, 0, 0, 0, 0, 0].**

**1.6.** Care afirmaţii sunt adevărate după execuţia algoritmul **detVect(n)? (n** întreg)**.** Algoritmul **detVect(n)** apelează algoritmul **caut(c,v,nr)** cu **c** natural**, v** vector, **nr** lungimea lui **v.** (Pseudocod; sursă: personală)

```
Algorithm detVect(n):
  For i← 1,10 execute
    v[i]← 0
  EndFor
  nr← 0
  If  n < 0
    then
           n← -n
  EndIf
  If n = 0
    then
           nr    ← nr+1
           v[nr] ← 0
    else
       While n > 0 execute
           p← 1
           m← n
           While m > 9 execute
                m← m DIV 10
                p← p*10
           EndWhile
           a← n DIV p
           If caut(a,v,nr) = 0
                then
                   nr   ← nr+1
                  v[nr]← a
           EndIf
           n← n MOD p
      EndWhile
  EndIf
EndAlgorithm
```

```
Algorithm caut(c,v,nr):
   For i← 1,nr EndFa
      If v[i] = c
            then
                  return 1
      EndIf
   EndFor
   return 0
EndAlgorithm
```

**A**. determină completarea vectorului **v** cu frecvenţa cifrelor lui **n**;
**B.** determină completarea vectorului **v** cu mulţimea cifrelor lui **n**, în ordinea apariţiei în **n, n≠0**;
**C.** determină completarea vectorului **v** cu mulţimea cifrelor nenule ale lui **n**, în ordinea apariţiei în **n, n≠0**;
**D**. afirmaţiile **A**., **B**., **C**. sunt false.

n=3012 => p=1000 a= 3 => v=[3], nr=1 =>
=> n=12 => p=10 a=1 => v[3,1], nr=2 =>
=> n=2 => p=1 a=2 => v[3,1,2], nr=3

**1.7.** Ce conţine vectorul v, după execuţia algoritmului **detVect(n)? (n** întreg). Algoritmul **detVect(n)** apelează algoritmul **caut(c,v,nr)** cu **c** natural, **v** vector şi **nr** lungimea lui **v**. (Pseudocod; sursă: personală)

```
Algorithm detVect(n,v,nr):
  For i← 1,10 execute
     v[i]← 0
  EndFor
  nr← 0;
  If (n < 0)
    then
          n← -n
  EndIf
  If n = 0
    then
          nr    ← nr+1
          v[nr]← 0
    else
     p← [lg(n)] {log in baza 10}
     p← 10^p
     While n > 0 execute
        a← n DIV p
        If caut(a,v,nr) = 0
          then
             nr  ← nr+1
             v[nr] ←a
        EndIf
       n← n MOD p
       p← p DIV 10
      EndWhile
  EndIf
EndAlgorithm
```

```
Algorithm caut(c,v,nr):
   For i← 1,nr execute
      If v[i] = c
         then
            return 1
       EndIf
   EndFor
   return 0
EndAlgorithm
```

**A.** Pentru **n= -123406** ⇒ **nr=5**, iar **v=[1,2,3,4,0,6];**
**B.** Pentru **n= - 45616** ⇒ **nr=0**, iar **v=[0,0,0,0,0,0,0,0,0,0];**
**C.** Pentru **n= 4561161** ⇒**nr=4**, iar **v=[4,5,6,1];**
**D.** Pentru **n= -123456** ⇒ **nr=6**, iar **v=[1,2,3,4,5,6].**

**1.8.** Ce conţine vectorul **v**, după execuţia algoritmului **detVect(n)? (n** întreg)**.** Algoritmul **detVect(n)** apelează algoritmul **caut(c,v,nr)** cu **v** vector şi **nr** lungimea lui **v**). (Pseudocod; sursă: personală)

```
Algorithm detVect(n):
  For i← 1,10 execute
    v[i]← 0
  EndFor
  nr← 0
  If n < 0
    then
           n← -n
  EndIf
  If n = 0
    then
        nr   ← nr+1
        v[nr]← 0
    else
       While n > 0 execute
          p← 1
          m← n
          While m > 0 execute
                 m← m DIV 10
                 p← p*10
          EndWhile
          p← p DIV 10
          a← n DIV p
          If caut(a,v,nr) = 0
              then
                 nr   ← nr+1
                 v[nr]← a
          EndIf
          n← n MOD p
       EndWhile
  EndIf
EndAlgorithm
```

```
Algorithm caut(c,v,nr):
   For i← 1,nr execute
      If v[i] = c
         then
            return 1
       EndIf
   EndFor
   return 0
EndAlgorithm
```

**A.** Pentru **n= 3025** => **nr=4**, iar vectorul **v=[3,0,2,5];**
**B.** Pentru **n= 4561** => **nr=4**, iar vectorul **v=[4,5,6,1];**
**C.** Pentru **n= -45616** => **nr=4**, iar vectorul **v=[4,5,6,1];**
**D.** Pentru **n= 6060** => **nr=1**, iar vectorul **v=[6].**

**1.9.** Care afirmaţii sunt adevărate după execuţia algoritmului **cifre(n, m)**? (**n** întreg, **m** cifră)
(Pseudocod; sursă: personală)

```
Algorithm cifre(n,m):
  For i← 0,9 execute
    v[i]← 0
  EndFor
  If n < 0
    then n← -n
  EndIf
  If n = 0
    then
     v[0]← 1
    else
    While n > 0 execute
       u← n MOD 10
       If (u MOD m = 0)
         then
           v[u]← 1
       EndIf
      n← n DIV 10
    EndWhile
  EndIf
EndAlgorithm
```

**A.** pentru **n= 1234**, **m=3** => **v=[0, 1, 1, 0, 1, 0, 0, 0, 0, 0];**
**B.** pentru **n= 9876**, **m=3** => **v=[0, 0, 0, 0, 0, 0, 1, 0, 0, 1];**
**C.** pentru **n= ⁻123456**, **m=2** => **v=[0, 0, -1, 0, -1, 0, -1, 0, 0, 0];**
**D.** pentru **n= ⁻ 9876543**, **m=4** => **v=[0, 0, 0, 0, 1, 0, 0, 0, 1, 0].**

**1.10.** Ce se returnează după execuţia algoritmului **f(n)? (n** natural**)** (Pseudocod; sursă: personală)

```
Algorithm f(n):
  If n < 10
    then
           return n
  EndIf
  u← n MOD 10
  p← f(n DIV 10)
  If u > p
    then
           return u
    else
           return p
  EndIf
EndAlgorithm
```

**A. n**, pentru **n=0**;
**B. 9**, pentru **n=798659**;
**C. 5**, pentru **n=598765**;
**D. 9**, pentru **n>0**.

**1.11.** Ce se returnează după execuţia algoritmului **f(n)? (n** natural**)** (Pseudocod; sursă: personală)

```
Algorithm f(n):
  If n < 10
    then
           return n
  EndIf
  u← n MOD 10
  p← f(n DIV 10)
  If u < p
    then
           return u
    else
           return p
  EndIf
EndAlgorithm
```

**A. 9**, pentru **n=798659**;
**B. 0**, pentru **n=0**;
**C. 5**, pentru **n=598765**;
**D. 0**, pentru **n>0.**

**1.12.** Ce se returnează după execuţia algoritmului **f(n)? (n** natural**)** (Pseudocod; sursă: personală)

```
Algoritmul f(n):
  If n < 10
    then
      If n MOD 2 = 0
         then
           return n
         else
           return -1
      EndIf
  EndIf
  If n MOD 2 = 0
    then
      u← n MOD 10
    else
      u← -1
  EndIf
  p← f(n DIV 10)
  If u < p
    then
           return u
    else
           return p
  EndIf
EndAlgorithm
```

**A. n**, pentru **n=0**;
**B. 1**, pentru **n=7135**;
**C. -1**, pentru **n=98764**;
**D. 2**, pentru **n=4628**.

**1.13.** Ce se returnează după execuţia algoritmului **f(n)? (n** natural**)** (Pseudocod; sursă: personală)

```
Algorithm f(n):
  If n < 10
    then
      If n MOD 2 > 0
         then
           return n
        else
           return -1
      EndIf
  EndIf
  If n MOD 2 > 0
    then
           u← n MOD 10
    else
           u← -1
  EndIf
  p← f(n DIV 10)
  If u < p
    then
         return u
    else
         return p
  EndIf
EndAlgorithm
```

**A. n**, pentru **n=0**;
**B. -1**, pentru **n=8468**;
**C. 5**, pentru **n=98765**;
**D. 1**, pentru **73137**.

**1.14.** Ce returnează algoritmul **f(n,neg,o)** pentru n număr întreg? (Pseudocod; sursă: personală)

```
Algorithm f(n,neg,o):
   If n < 0
     then
       n  ← -n
       neg←1
   EndIf
   If n > 0
     then
        return f(n DIV 10, neg, o*10+ (n MOD 10))
   EndIf
   If neg = 1
     then
       return –o
     else
       return  o
   EndIf
EndAlgorithm
```

**A. n** pentru apelul **f (0, 0, 0);**
**B. 111** pentru apelul **f (111, 0, 0);**
**C. - 222** pentru apelul **f (-222, 0, 0);**
**D. 213** pentru apelul **f (321, 0, 0).**

**1.15.** Ce returnează algoritmul **f(n,c1,c2)** pentru **n,c1,c2** numere naturale? (Pseudocod; sursă: personală)

```
Algorithm f(n, c1, c2):
  p  ← 1
  nou← 0
  While n > 0 execute
     u← n MOD 10
     If u = c1
        then
           nou← nou+c2*p
        else
           nou← nou+u*p
     EndIf
     p← p*10
     n← n DIV 10
  EndWhile
  return nou
EndAlgorithm
```

**A. n,** pentru apelul **f (n, 0, 0);**
**B. 1212** pentru apelul **f (2121, 1, 2);**
**C. 3234** pentru apelul **f (1214, 1, 3);**
**D. 2232** pentru apelul **f (1131, 2, 1).**

**1.16.** Ce se returnează? (Pseudocod; sursă: personală)

```
Algorithm ceFace(n, p):
  If n > 0
    then
       If ((n MOD 10) MOD 3) = 0
          then
               return (9-(n MOD 10))*p + ceFace(n DIV 10,p*10)
          else
               return ceFace(n DIV 10,p)
       EndIf
    else
       return 0
  EndIf
EndAlgorithm
```

**A. 6** pentru apelul **ceFace (12345, 10)**;
**B. 630** pentru apelul **ceFace (123456, 10)**;
**C. 6300** pentru apelul **ceFace (1234567, 10);**
**D. 6300** pentru apelul **ceFace (123456789, 10).**

**1.17.** Ce se returnează după execuţia algoritmului **ceFace(n), n** natural nenul?
(Pseudocod; sursă: personală)

```
Algorithm ceFace(n):
  p← 1
  m← 0
  While n > 0 execute
     u← n MOD 10
     If u MOD 2 = 0
        then
           m← m + u*p
           p← p*10
     EndIf
     n←n DIV 10
  EndWhile
  return m
EndAlgorithm
```

**A. 6** pentru apelul **ceFace(12345);**
**B. 642** pentru apelul **ceFace(123456);**
**C. 1357** pentru apelul **ceFace(1234567);**
**D. 2468** pentru apelul **ceFace(123456789).**

**1.18.** Ce se returnează după execuţia algoritmului **ceFace(n,c), n,c** naturale? (Pseudocod; sursă: personală)

```
Algorithm ceFace(n,c):
  p← 1
  m← 0
  While n > 0 execute
     u← n MOD 10
     If u > c
        then
           m← m+u*p
           p← p*10
     EndIf
     n← n DIV 10
  EndWhile
  return m
EndAlgorithm
```

**A. 6** pentru apelul **ceFace(12346,4);**
**B. 3456** pentru apelul **ceFace(123456,2);**
**C. 1357** pentru apelul **ceFace(1234567,1);**
**D. 798** pentru apelul **ceFace(1274968,6).**

**1.19.** Algoritmul **pm (n)** apelează algoritmul **pr(n), n** întreg. Ce afişează algoritmul **pm**, pentru valorile parametrului **n** ? (Pseudocod; sursă: personală).

```
Algorithm pm(n):
  p ← pr(n)
  Afisare p
  While p ≠ n execute
    p← pr(p)
    Write  p
  EndWhile
EndAlgorithm
```

```
Algorithm pr (n):
  neg← 0
  If n < 0
    then
          neg← 1
          n  ← -n
  EndIf
  nr← [lg(n)]
  p ←  10^nr
  c ← n DIV p
  r ← n MOD p
  If neg = 1
    then
          return -(r*10+c)
  else
          return  (r*10+c)
  EndIf
EndAlgorithm
```

**A.** pentru **n = 1357** se afişează **7135 5713 3571 1357**;
**B.** pentru **n = 4567** se afişează **5674 6745 7456 4567**;
**C.** pentru **n = -123** se afişează **-231 -312 -123**;
**D.** pentru **n = 6789** se afişează **6789 7896 8967 9678.**

**1.20.** Ce returnează algoritmul **a(n)**, n natural? (Pseudocod; sursă: personală).

```
Algorithm a(n):
  If n = 0
     then
           return 0
  EndIf
  For i← 0,9 execute
      v[i]← 0
  EndFor
  While n > 0 execute
        v[n MOD 10] ← v[n MOD 10]+1
        n           ← n DIV 10
  EndWhile
  c← 1
  While (c < 10) AND (v[c] = 0) execute
     c← c+1
  EndWhile
  m   ← c
  v[c]← v[c]-1
  For c←0,9 execute
    While v[c] > 0 execute
          m   ← m*10+c
          v[c]← v[c]-1
    EndWhile
  EndFor
  return m
EndAlgorithm
```

**A**. pentru **n = 13507** se returnează **75310;**
**B**. pentru **n = 400123** se returnează **1234**;
**C**. pentru **n = 1203** se returnează **3021;**
**D**. pentru **n = 4054022** se returnează **2002445**.

**1.21.** Algoritmul **ab(a,b)** apelează algortimul **x(c,n)** , **a,b** sunt date şi naturale. Verificaţi care din afirmaţiile de mai jos sunt corecte. (Pseudocod; sursă: personală).

```
Algorithm ab(a, b):
  While a>0 execute
      u ← a MOD 10
      If x(u,b) = 0
        then
          return 0
      EndIf
      a← a DIV 10
  EndWhile
  return 1
EndAlgorithm
```

```
Algorithm x(c, n):
  While n>0 execute
     If c = (n MOD 10)
       then
           return  1
     EndIf
     n← n DIV 10
  EndWhile
  return 0
EndAlgorithm
```

**A.** Apelul **ab(1234, 5678)** returnează **1**;
**B.** Apelul **ab(1234, 34152)** returnează **1**;
**C.** Apelul **ab(1234, 47612)** returnează **0**;
**D.** Apelul **ab(1111, 1212)** returnează **1.**

**1.22.** Care algoritmi determină primalitatea lui **n**, întreg: **true** dacă **n** e prim, **false** altfel?
(Pseudocod; sursă: personală).

```
1)
Algorithm prim(n):
  If (n > 2) AND (n MOD 2 = 0)
     then
           return false
  EndIf
  d← 3
  While d*d ≤ n execute
     If (n MOD d) = 0
         then
               return false
     EndIf
     d← d+2
  EndWhile
  return true
EndAlgorithm
```

```
2)
Algorithm prim(n):
  If n ≤ 1
    then
           return false
  EndIf
  If (n>2) AND ((n MOD 2) = 0)
     then
           return false
  EndIf
  d← 3
  While d*d ≤ n execute
     If (n MOD d) = 0
        then
           return false
     EndIf
     d← d+2
  EndWhile
  return true
SfAlgorithm
```

```
3)    {la apelul primar, d=2}
Algorithm prim(n, d):
  If n ≤ 1
    then
           return false
  EndIf
  If d*d > n
    then
           return true
  EndIf
  If (n MOD d) = 0
    then
           return false
  EndIf
  If d = 2
    then
       return prim(n,d+1)
    else
       return prim(n,d+2)
  EndIf
EndAlgorithm
```

```
4)                {la apelul primar d=2}
Algorithm prim(n,d):
  If n ≤ 1
    then
       return false
  EndIf
  d← 2
  If d*d > n
    then
       return true
  EndIf
  If (n MOD d) = 0
    then
       return false
  EndIf
  return   prim(n,d+2)
EndAlgorithm
```

A. **1,2**;
B. **1,2,3**;
C. **2,3**;
D. **2,3,4**;

**1.23.** Algoritmul **a(n)** cu parametrul **n≥3**, natural, iar **v** este vector. Ce va conţine vectorul **v** după execuţia algoritmului **a(n)**? (Pseudocod; sursă: personală).

```
Algorithm a(n):
  v[1]← 2
  nr  ← 1
  m   ← 3
  While m ≤ n execute
      radM← [√m]
      i   ← 1
      While (i < nr) AND (v[i] ≤ radM) execute
            If (m MOD v[i]) = 0
                  then
                        i← nr+1
                  else
                        i← i+1
            EndIf
      EndWhile
      If i ≤ nr
            then
                  nr   ← nr+1
                  v[nr]← m
      EndIf
      m← m+2
  EndWhile
EndAlgorithm
```

**A.** pentru **n= 7 => v = [2, 3, 5, 7, 9, 11, 13]**
**B.** pentru **n= 5 => v = [2, 3, 5, 7, 9]**
**C.** pentru **n=20 => v = [2, 3, 5, 7, 11, 13, 17, 19]**
**D.** pentru **n=20 => v = [2, 3, 5, 7, 9, 11, 13, 15, 17, 19]**

**1.24.** Care afirmaţii sunt adevărate la apelul programului **p(a,b)**? (**a**,**b** naturale, nenule)
(Pseudocod; sursă: Politehnica Bucureşti).

```
Algorithm p(a,b):
  If a = 0
     then
          return 0
  EndIf
  If (b MOD a) = 0
      then
          return p(a-1, b) + 1
      else
          return p(a-1, b)
   EndIf
EndAlgorithm
```

**A.** dacă **p(a,2) = 1** verifică că **a** este prim, **a>1**;
**B.** dacă **p(a-1,a) = 1** verifică că **a** este prim, **a>1**;
**C.** dacă **p(b DIV 2, b) = 1** verifică că **b** este prim, **b>1**;
**D.** dacă **p(a*a,a) = 2** verifică că **a** este prim, **a>1**.

**1.25.** Ce conţine vectorul **v**, după execuţia algoritmului **a(n), n** natural, **n>0** ?
(Pseudocod; sursă: personală).

```
Algorithm a(n):
   c← 0
   d← 2
   While d*d < n execute
      If (n MOD d) = 0
           then
                c    ← c+1
                v[c] ← d
                c    ← c+1
                v[c] ← n DIV d
      EndIf
      d← d+1
   EndWhile
   If d*d = n
           then
                c    ←c+1
                v[c] ←d
   EndIf
EndAlgorithm
```

**A.** pentru **n = 100 => v = [1, 100, 2, 50, 4, 25, 5 ,20, 10];**
**B.** pentru **n = 36 => v = [1, 2, 3, 4, 9, 12, 18, 36, 6];**
**C.** pentru **n = 64 => v = [2 ,4, 16, 32, 8];**
**D.** pentru **n = 144 => v = [2, 72, 3, 48, 4, 36, 6, 24, 8, 18, 12].**

**1.26.** Care afirmaţii sunt adevărate după execuţia algoritmului **s(n,d)** ? (Pseudocod; sursă: personală).

```
Algorithm s(n,d):
  If n ≤ 1
    then
          return 0
  EndIf
  If d*d > n
    then
          return –n
  EndIf
  If d*d = n
    then
          return d – n
  EndIf
  If (n MOD d) = 0
    then
          return (d + (n DIV d)) + s(n,d+1)
    else
          return s(n,d+1)
  EndIf
EndAlgorithm
```

**A.** pentru **n** natural, impar şi **d=2** se returnează un număr negativ;
**B.** pentru **n** natural, prim şi **d=2** se returnează **–n**;
**C.** pentru **n** natural, prim şi **d=1** se returnează **–n**;
**D.** pentru **n** natural, **n>1**, număr perfect şi **d=1** se returnează **n**;

**1.27.** Care afirmaţii sunt adevărate după execuţia algoritmului **s(n,d)**? **n** natural, nenul. (Pseudocod; sursă: personală).

```
Algorithm s(n,d):
  If n = 1
    then
          return 1
  EndIf
  If d*d = n
    then
          return d
  EndIf
  If d*d > n
    then
          return 0
  EndIf
  If (n MOD d) = 0
    then
          return d + (n DIV d) + s(n,d+1)
    else
          return s(n,d+1)
  EndIf
EndAlgorithm
```

**A.** pentru **n∈{16, 25}** şi **d=1** se returnează **31**;
**B.** pentru **n∈{14, 15, 23}** şi **d=1** se returnează **24**;
**C.** pentru **n∈{20, 26, 41}** şi **d=1** se returnează **42**;
**D.** pentru **n∈{24, 36, 59}** şi **d=1** se returneaza **60**.

**1.28.** Ce returnează algoritmul următor? (**n,i** naturale nenule). (Pseudocod; sursă: personală).

```
Algorithm ceFace(n, i):
      If i * i > n
            then
                  return  1
      EndIf
      If i * i = n
            then
                  return  i + ceFace(n,i+1)
      EndIf
      If (n MOD i) = 0
            then
                  return  i + (n DIV i) + ceFace(n,i+1)
            else
                  return  ceFace(n,i+1)
      EndIf
SfAlgorithm
```

**A.** Apelul **ceFace(10,2)** returnează **9**;
**B.** Apelul **ceFace(12,1)** returnează **29**;
**C.** Apelul **ceFace(28,2)** returnează **28**;
**D.** Apelul **ceFace(496,2)** returnează **496**.

**1.29.** Numărul **30 = 2*3*5** are **4** divizori impari **{1, 3, 5, 15}** şi **4** divizori pari **{2, 6, 10, 30}.**
Care afirmaţii sunt adevărate? (sursă: personală).

**A.** **4500** are **8** divizori impari;
**B.** **4500** are **12** divizori impari şi **24** divizori pari;
**C.** **3600** are **7** divizori impari şi **38** divizori pari;
**D.** **3600** are **9** divizori impari **şi 36** divizori pari.

**1.30.** Numărul **3** poate fi scris ca sumă de numere naturale **> 0** astfel**:**
**3 = 1+1+1 = 1+2 = 2+1 = 3**
Care afirmaţii sunt false?
(sursă: problema 16, Ingeniozitate şi surpriză în matematică, Charles W. Trigg, Orizonturi, 1975).

**A.** Numărul **10** se scrie în **1023** moduri;
**B.** Numărul **8** se scrie în **128** moduri;
**C.** Numărul **8** se scrie în **127** de moduri;
**D.** Numărul $\mathbf{2^{16}}$ se scrie în **32768** de moduri.

**1.31.** Care dintre cei **4** algoritmi determină cel mai mic multiplu comun dintre **a** şi **b** (naturale şi nenule) ?

```
1.
Algorithm  cmmmc (a, b):
  If a ≤ b
    then
           a← a-b
           b← a+b
           a← b-a
  EndIf
  m← a
  While (m MOD b) > 0 execute
           m← m+a
  EndWhile
  return m
EndAlgorithm
```

```
2.
Algorithm  cmmmc (a, b):
  If a ≤ b
    then
           a← a*b
           b← a+b
           a← b-a
  EndIf
  m← a
  While  (m MOD b) > 0 execute
           m← m+a
  EndWhile
  return m
EndAlgorithm
```

```
3.
Algorithm  cmmmc (a, b):
  If a ≤ b
    then
           a← a DIV b
           b← a*b
           a← b DIV a
  EndIf
  m← a
  While (m MOD a) > 0 execute
           m← m+b
  EndWhile
  return m
EndAlgorithm
```

```
4.
Algorithm  cmmmc (a, b):
  If a > b
    then
           m← a
    else
           m← b
  EndIf
  If m = a
    then
           While (m MOD b) > 0 execute
                 m← m+a
           EndWhile
    else
           While (m MOD a) > 0 execute
                 m← m+b
           EndWhile
  EndIf
  return m
EndAlgorithm
```

**A. 1,2,3;**
**B. 1,2;**
**C. 1,4;**
**D. 1,2,3,4.**

**1.32.** Ce valori vor avea **a**, **b** şi **k** la sfârşitul secvenţei? (sursă: concurs/admitere la FMI, UBB Cluj-Napoca)

```
a← 20
b← 90
k← 0
If a < b
  then
     b ← b*a
     a ← b DIV a
     b ← b DIV a
EndIf
While a ≥ b execute
   a ← a-b
   k ← 2*k+1
EndWhile
```

**A. 90, 70, 7;**
**B. 90, 20, 5;**
**C. 10, 20, 15;**
**D. 10, 30, 3.**

**1.33.** Care afirmaţii sunt adevărate după execuţia algoritmului **sp(n)** pentru n întreg? (Pseudocod; sursa: personală)

```
Algorithm sp(n):
  For i←2,n   execute
    c← 0
    While (n MOD i) = 0 execute
        n← n DIV i
        c← c+1
    EndWhile
    If c > 0
      then
          Write i," ",c
    EndIf
  EndFor
EndAlgorithm
```

**A.** Afişează perechi de numere prime (**n≥2**);
**B.** Afişează toţi divizorii lui **n** şi puterile lor (**n≥2**);
**C.** Afişează factorii primi ai lui n şi puterile lor (**n≥2**);
**D.** Algoritmul nu afişează nimic, dacă **n<2**.

**1.34.** Fie secvenţa de instrucţiuni. Care afirmaţii sunt adevărate? (Pseudocod; sursa: personală)

```
Read a,b {numere întregi}
x← 1
While (a>0) AND (b>0) execute
   If (a MOD 10) < (b MOD 10)
     then
          x← 0
   EndIf
   a← a DIV 10
   b← b DIV 10
EndWhile
If (x=1) AND (b=0)
  then
          Write ”DA”
  else
          Write ”NU”
EndIf
```

**A.** Afişează **„NU”** pentru perechi de numere strict negative**, a,b<0;**
**B.** Afişează **„DA”** dacă **a<b;**
**C.** Afişează **„DA”** dacă fiecare cifră a lui **a este ≥** decât cifra corespondentă a lui **b (unităţi, zeci, etc.)** şi **nrCif(b)< nrCif(a), a,b>0;**
**D.** Afişează **„DA”** pentru **a=534, b=112.**

**1.35.** Fie secvenţa de instrucţiuni. Care variante sunt corecte? (Pseudocod; sursa: personală)

```
Read n     {numar natural}
a← n MOD 10
While n > 9 execute
     n← n DIV 10
EndWhile
If (a MOD 2) = (n MOD 2)
  then
          Write ”DA”
  else
          Write ”NU”
EndIf
```

**A.** Afişează „**DA**” pentru **n ≥ 1**;
**B.** Afişează „**DA**” dacă ultima cifră a lui n este egală cu penultima cifră a lui n (**n>9** şi **n<100**);
**C.** Afişează „**NU**” dacă prima cifră a lui **n** are aceeaşi paritate cu ultima cifră a lui **n**, **n≥ 1**;
**D.** Afişează „**DA**” pentru **n=1233**.

**1.36.** Ce face secvenţa de instrucţiuni pentru şirul: **15, 24, 35, 25, 75, 26, 0**. (sursă: concurs/admitere la FMI, UBB Cluj-Napoca)

```
Read x {x natural}
nr←0
s←0
While  x ≠ 0 execute
  nr← nr+1
  If ((nr MOD 2) = 0) AND ((x MOD 5) = 0)
     atunci
           s← s + x MOD 10
  EndIf
  Read x
EndWhile
Write  s," ", nr
```

**A.** Afişează **15 7;**
**B.** Afişează **20 5;**
**C.** Afişează **5 6;**
**D.** Afişează suma dintre ultimele cifre ale numerelor de indice par din şirul citit, şi numărul de elemente citite**.**

**1.37.** Ce face secvenţa? (sursă: personală şi concurs/admitere la FMI, UBB Cluj-Napoca)

```
Read n      {n>0}
i← 1
While n > 0 execute
  If (n MOD 2) > 0
    then
          Write i
  EndIf
  i← i + 1
  n← n DIV 2
EndWhile
```

**A.** Afişează secvenţa: **12345** pentru **n=31**;
**B.** Afişează secvenţa: **234** pentru **n=14**;
**C.** Afişează **1** la începutul secvenţei, pentru **n** impar, **n>**0;
**D.** Afişează două cifre pentru $\mathbf{n=2^k}$**.**

**1.38.** Ce returnează algoritmul **f(n)** pentru **n** număr natural? (Pseudocod; sursă: personală)

```
Algorithm f(n):
  If n < 10
    then
      If (n MOD 2) = 0
        then
                return n
       else
                return -1
      EndIf
  EndIf
  If (n MOD 2) = 0
    then
          u← n MOD 10
    else
          u← -1
  EndIf
  p← f(n DIV 10)
  If u > p
    then
          return u
    else
          return p
  EndIf
EndAlgorithm
```

**A. n,** pentru **n=0;**
**B. -1,** pentru **n=7135;**
**C. 8,** pentru **n=98764;**
**D. 2,** pentru **n=4628.**

**1.39.** Ce afişează secvenţa? (sursă: personală)

```
Read n {număr natural ≥ 0}
t← 1
c← n MOD 10
n← n DIV 10
While (t = 1) AND (n > 0) execute
  If (n MOD 10) > c
    then
           t← 0
  EndIf
  c← n MOD 10
  n← n DIV 10
EndWhile
```

**A.** Afişează **1,** dacă **n>0;**
**B.** Afişează **0,** dacă **n=1234;**
**C.** Afişează **0,** dacă **n=4321;**
**D.** Afişează **1,** dacă **n=55555.**

**1.40.** Ce afişează secvenţa? (sursă: personală)

```
Read x,m          {x,m întregi}
y←1
While m > 0 execute
  If (m MOD 2) = 0
    then
           m← m DIV 2
           x← x*x
    else
           m← m-1
           y← y*x
  EndIf
EndWhile
Write y
```

**A.** Afişează $x^2$**,** dacă **m=2;**
**B.** Afişează **1,** pentru **m <0;**
**C.** Afişează $x^k$**,** dacă $m=2^k$**;**
**D.** Afişează un număr negativ dacă **x<0.**

**1.41.** Fie secvenţa următoare, cu **n** este natural, nenul? Ce se afişează? (sursă: personală)

```
Read n
p← 2
s← 0
While p ≤ n execute
    s← s + (n DIV p)
    p← p*2
EndWhile
Write s
```

**A.** suma numerelor divizibile cu **2 şi 4** şi mai mici ca **n;**
**B.** suma numerelor divizibile cu **4;**
**C.** exponentul lui **2** (ca factor) în **descompunerea lui n!;**
**D.** exponentul lui **4** (ca factor) în **descompunerea lui n!.**

**1.42.** Care din următoarele propoziţii sunt adevărate? (sursă: personală)

```
Read n {natural cu cel mult 9 cifre}
While  n ≥ 10  execute
      s← 0 )
      While n ≠ 0 execute
            s← s + (n MOD 10)
            n← n DIV 10
      EndWhile
      n← s
EndWhile
Write n
```

**A.** corpul ciclului exterior se execute de **n** ori;
**B.** indiferent de valoarea lui **n**, se afişează cifra **9**;
**C.** indiferent de valoarea lui **n**, se afişează o cifră;
**D.** corpul ciclului exterior se execute de cel mult **3** ori.

**1.43.** Ce se afişează? (sursă: personală)

```
 Read n  {natural,n>1}
 d← 2
 While (n MOD d) ≠ 0 execute
      d← d+1
 EndWhile
 While (n MOD d) = 0 execute
      n← n DIV d
 EndWhile
 If n = 1
    then
           Write d
    else
           Write n
 EndIf
```

**A**. valoarea iniţială a lui **n**, dacă **n** este prim;
**B**. **1**, dacă **n** este prim;
**C**. **2**, dacă **n** este $n=2^k$;
**D**. valoarea iniţială a lui **n**, dacă valoarea iniţială a lui **n** nu este număr prim.

**1.44.** Ce se afişează (**n** întreg)? (sursă: personală)

```
Read n
i← 2
p← 1
While n > 1 execute
  k← 0
  While  (n MOD i) = 0  execute
      k← i
      n← n DIV i
  EndWhile
  If k ≠ 0
    then
           p← p*k
           i← i+1
  EndIf
EndWhile
Write p
```

**A.** **1**, pentru **n=-6**;
**B.** **6**, dacă **n=6**;
**C.** **n!,** dacă **n>0**;
**D.** produsul factorilor primi ai lui **n**, **n** natural **n>1**.

**1.45.** Care e valoarea lui **z** după execuţia secvenţei? (sursă: personală)

```
Read n {n număr natural}
z← 0
While n > 0 execute
      c← n MOD 10
      n← n DIV 10
      If (c MOD 2) = 0
        then
            z← z*10 + c
  EndIf
EndWhile
Write z
```

**A.** Dacă **n=12345**, **z** este **12345**;
**B.** Dacă **n=54321**, z este **531**;
**C.** Dacă **n=135**, **z=0**;
**D.** Dacă **n=16345**, **z=46.**

**1.46.** Fie secvenţa următoare. Care afirmaţii sunt adevărate? (sursă: personală)

```
Read n,c {n natural nenul, c cifra}
z← 0
While (n MOD 10) = c  execute
      n← n DIV 10
|     z← z+1
EndWhile
Write z
```

**A.** Afişează **3**, pentru **n=123** şi **c=3**;
**B.** Afişează **2**, pentru **n=12003** şi **c=0**;
**C.** Afişează **4**, pentru **n=1277771** şi **c=7**;
**D.** Afişează **3**, pentru **n=12555** şi **c=5**.

**1.47.** Fie secvenţa următoare. Care afirmaţii sunt adevărate? (sursă: personală)

```
Read n,k    {n,k naturale nenule}
z← 0
For i← 1,n  execute
  If (i MOD k) = 0
    then
            z← z+i
  EndIf
EndFor
Write z
```

**A.** Afişează **1** pentru **n>k**;
**B.** Afişează $k * \left(1 + \left[\frac{n}{k}\right]\right) * \left[\frac{n}{k}\right] / 2$ pentru **n>k**;
**C.** Afişează **105** pentru **n=40** şi **k=7**;
**D.** Afişează o putere a lui **2**, pentru **n>k=2**.

**1.48.** Ce valoare returnează algoritmul **a(n,d)** ? (sursă: personală)

```
Algorithm a (n, d):          {n, d întregi}
  If d*d > n
    then
          return 1+n
  EndIf
  If d*d = n
    then
          return  d + a(n,d+1)
  EndIf
  If (n MOD d) = 0
    then
          return  a(n,d+1)
    else
          return d + (n DIV d) + a(n,d+1)
EndAlgorithm
```

**A**. **-4**, pentru **a(-5,1);**
**B.** **4**, pentru **a(3,2);**
**C**. **6**, pentru **a(5,2);**
**D**. Suma divizorilor proprii, pentru **a(n,2)** cu **n>1**.

**1.49.** Ce returnează algoritmul următor? (sursă: personală)

```
Algorithm a(b,c):           {b,c naturale}
  While (c>0) AND (b ≠ c MOD 10) execute
     c← c DIV 10
  EndWhile
  If (c = 0)
    then
          return false
    else
          return true
  EndIf
EndAlgorithm
```

**A. false**, pentru **a(1,1);**
**B. true**, pentru **a(2,2);**
**C. false**, pentru **a(5,22);**
**D. true**, pentru **a(5,25).**

**1.50.** Ce returnează algoritmul **a(n,p)**? (sursă: personală)

```
Algorithm a(n,p):     {n,p naturale}
  If n > 0
    then
      If ((n MOD 10) MOD 4) = 0
        then
                return (10-(n MOD 10))*p + a(n DIV 10, p*10)
        else
                return a(n DIV 10, p)
      EndIf
  EndIf
  return  0
EndAlgorithm
```

**A.** **14**, pentru **n=147**, **p=1**;
**B.** **62**, pentru **n=451823**, **p=1**;
**C.** **47**, pentru **n=512346**, **p=1**;
**D.** **60**, pentru **n=632514**, **p=10**.

**1.51.** Se consideră numerele naturale **m** și **n (0 ≤ m ≤ 10, 0 ≤ n ≤ 10)** și algoritmul **ack(m, n)** care calculează valoarea funcției **Ackermann** pentru valorile **m** și **n**. (sursă: concurs/admitere la FMI, UBB Cluj-Napoca)

```
Algorithm ack(m, n):
    If m = 0
      then
           return n + 1
      else
         If (m > 0) AND (n = 0)
           then
                return ack(m - 1, 1)
           else
                return ack(m - 1, ack(m, n - 1))
      EndIf
    EndIf
EndAlgorithm
```

Precizați de câte ori se autoapelează algoritmul **ack(m, n)** prin EndFarea secvenței de instrucțiuni:
**m ← 1, n ← 2**
**ack (m, n)**

**A.** de **7** ori;
**B.** de **10** ori;
**C.** de **5** ori;
**D.** de același număr de ori ca și în cazul executerii secvenței de instrucțiuni:
**m ← 1, n ← 3**
**ack (m, n)**

**1.52**. Ce conţine vectorul **v**, după execuţia secvenţei, **n** întreg? (sursă: personală)

```
Read(n)
For c← 0,9 execute
  v[c]← 0
EndFor
While n>0 execute
  c      ← n MOD 10
  v[c]   ← v[c] + 1
  n      ← n DIV 10
EndWhile
```

**A.** doar cifre binare, pentru **n>0;**
**B.** doar cifre **0,** pentru numere strict negative**;**
**C.** frecvenţa de apariţie a cifrelor, pentru **n>0;**
**D.** doar frecvenţa de apariţie a cifrei **0,** pentru **n>0.**

**1.53.** Ce returnează algoritmul **a(n)**, **n** natural nenul. (sursă: personală)

```
Algorithm a(n):
  For i← 0,9 execute
     x[i]← 0
  EndFor
  While n > 0 execute
      u     ← n MOD 10
      x[u]  ← x[u]+1
      n← n DIV 10
  EndWhile
  If x[0] > 0
     then
       return false
  EndIf
  For i← 1,9 execute
      If x[i] > 1
        then
           return false
      EndIf
  EndFor
  return  true
EndAlgorithm
```

**A. false,** dacă **n** conţine cifra **0;**
**B. true,** dacă **n** conţine toate cifrele nenule**;**
**C. false** dacă **n** conţine cifra **5** de două ori**;**
**D. true,** dacă **n** are doar cifre nenule distincte**.**

**1.54.** Ce returnează algoritmul? Se consideră subalgoritmul **a(x, b)** cu parametrii de intrare două numere naturale **x** și **b** (**$1 \le x \le 1000$, $1 < b \le 10$**). (sursă: concurs/admitere la FMI, UBB Cluj-Napoca).

```
Algorithm  a(x,b):
  s ← 0
  While x > 0 execute
     s ← s + x MOD b
     x ← x DIV b
  EndWhile
  return (s MOD (b - 1)) = 0
SfAlgorithm
```

**A.** suma cifrelor reprezentării în baza **b** a numărului natural **x**;
**B.** suma cifrelor reprezentării în baza **b-1** a numărului x dacă este divizibilă cu **b-1**;
**C.** true dacă numărul natural **x** este divizibil cu **b-1**;
**D.** true dacă suma cifrelor reprezentării în baza **b** a numărului **x** este divizibilă cu **b-1**.

**1.55.** Fie algoritmul **Calcul(a,b)** cu parametrii **a**,**b** numere naturale, **1 ≤ a,b ≤ 1000**. (sursă: concurs/admitere la FMI, UBB Cluj-Napoca).

```
1.  Algorithm Calcul (a,b):
2.    If a ≠ 0
3.      atunci
4.         return Calcul(a DIV 2, b+b) + b*(a MOD 2)
5.    EndIf
6.    return 0
7.  EndAlgorithm
```

Care afirmaţii sunt false?

- **A.** If **a** şi **b** sunt egale, algoritmul returnează valoarea lui **a**;
- **B.** If **a**=**1000** şi **b**=**2**, algoritmul se autoapelează de **10** ori;
- **C.** Valoarea calculată şi returnată de algoritm este **a** / **2** + **2*b**;
- **D.** Instrucţiunea de pe linia **6** nu se execute niciodată.

**1.56.** Ce returnează algoritmul recursiv **a(n,d)**? (**n,d** întregi) (sursă: personală)

```
Algorithm a(n,d)
  If  n < 2
    then
      return 0
  EndIf
  If d ≥ 2
    then
      If n MOD d = 0
        then
           return 0
        else
           return A(n,d-1)
      EndIf
  EndIf
  return 1
EndAlgorithm
```

- **A.** **0** pentru **A(-5,10)**;
- **B.** **0** pentru **A(2,2)**;
- **C.** **1** pentru **A(13,13)**;
- **D.** **0** pentru **A(14,2)**.

**1.57.** Ce face programul **pr(n)**, **n** natural, **n≥3**? (sursă: personală)

```
Algorithm pr(n):
  cont← 0
  i   ← 3
  While cont < n execute
    p← i DIV 2
    If i MOD 2 = 0
      then
           limJ← p*p-1
      else
           limJ← 2*p*(p+1)
    EndIf
    j← i+1
    While (cont < n) AND (j ≤ limJ) execute
      k← [√(i*i+j*j)]
      If  k*k = i*i+j*j
        then
           cont← cont+1
           Write cont, ".", i, j, k

      EndIf
      j← j+1
    EndWhile
    i← i+1
  EndWhile
EndAlgorithm
```

**A.** pt. **n=7** afişează:

1. 3 4 5
2. 6 8 10
3. 8 15 17
4. 9 12 15
5. 9 40 41
6. 10 24 26
7. 11 60 61

**B.** pt. **n=7** afişează:

1. 3 4 5
2. 5 12 13
3. 6 8 10
4. 7 24 25
5. 8 15 17
6. 9 12 15
7. 9 40 41

**C.** pt. **n=7** afişează:

1. 3 4 5
2. 5 12 13
3. 6 8 10
4. 7 24 25
5. 8 15 17
6. 9 12 15
7. 10 24 26

**D.** pt. **n=7** se afişează:

1. 3 4 5
2. 6 8 10
3. 8 15 17
4. 10 24 26
5. 11 60 61
6. 12 16 20
7. 12 35 37

**1.58.** Ce returnează algoritmul recursiv **a(n,d)**, **n,d** întregi? (sursă: personală)

```
Algorithm a(n,d):
 If n < 2
    then
       return 0
 EndIf
 If  d*d ≤ n
    then
      If n MOD d = 0
         then
            return 0
      EndIf
      If d = 2
        then
           return A(n,d+1)
        else
           return A(n,d+2);
      EndIf
 EndIf
 return  1
EndAlgorithm
```

A. **0** pentru **a (-5,10);**
B. **0** pentru **a (2,2);**
C. **1** pentru **a (13,4);**
D. **0** pentru **a (14,2).**

**1.59.** Care afirmaţii sunt adevărate după execuţia algoritmului următor? (sursă: personală)

```
Algorithm a(n,d)   {n,d naturali, n≥d, apel primar cu d=2}
  If d*d > n
    then
       Write  "1,", n
    else
       If  n MOD d =0
          then
            If n DIV d = d
                 then
                      Write d,","
                      a(n,d+1)
                 else
                      Write d, "," n DIV d, ","
                      a(n,d+1)
            EndIf
          else
                 a(n,d+1)
       EndIf
  EndIf
EndAlgorithm
```

**A.** Afişează divizorii naturali ai lui **n (n>1);**
**B.** Afişează numai divizorii proprii ai lui **n (n>1)**;
**C.** Afişează un număr par de divizori ai lui **n (n>1)**;
**D.** Afişează un număr impar de divizori ai lui **n (n>1)**.

**1.60.** Secvenţa următoare citeşte 3 numere întregi (a,b,c) şi le prelucrează. Care afirmaţii sunt adevărate? (sursă: personală)

```
Read a,b,c
If  a > b
  then
    a ←→ b {interschimbă valorile}
EndIf
If  b > c
  then
    b ←→ c {interschimbă valorile}
EndIf
If  a > b
  then
    a ←→ b {interschimbă valorile}
EndIf
Write a,b,c
```

**A.** Valoarea maximă dintre **a**,**b** şi **c** se va afişa pe prima poziţie;
**B.** Valoarea minimă dintre **a**,**b** şi **c** se va afişa pe ultima poziţie;
**C.** Valorile sunt afişate sortate descrescătoare;
**D.** Nici un răspuns din cele de mai sus nu e corect.

**1.61**. **Care afirmaţie e adevărată?** (sursă: personală)

```
Algorithm ceFace(n, L, i, j):
          {n>0,intreg, L,i,j naturale}
    nou ← 0
    p10 ← 1
    p   ← L-i+1
    cont← 1
    While  n > 0    execute
        c← n MOD 10
        If cont = p
           then
                nou← nou + j * p10
           else
                nou← nou + c * p10
        EndIf
        p10 ← p10 * 10
        n   ← n DIV 10
        cont← cont+1
    EndWhile
    return nou
EndAlgorithm
```

**A. ceFace(123,3,4,6),** retur **123**
**B. ceFace(654321,6,3,0),** retur **354321**
**C. ceFace(654321,6,3,1),** retur **651321**
**D. ceFace(654321,8,3,0),** retur **54321**

**1.62.** Care numere naturale au un singur divizor propriu? (sursă: personală)

**A.** Numerele prime;
**B.** Numerele impare;
**C.** Pătratele perfecte;
**D.** Pătratele perfecte $\mathbf{n=p^2}$, cu **p** prim.

**1.63.** Câte numere care au mai mult de **4 (>4)** divizori sunt de la **1** până la **n**, (**n≥12**) ? (sursă: personală)

**A.** $\mathbf{n - (1 + |\{p / p \le n, p \text{ prim}\}| + |\{p / p^2 \le n, p \text{ prim}\}| + |\{p / p^3 \le n, p \text{ prim}\}| + |\{pq / pq \le n, p,q \text{ prime}\}|)}$;
**B**. **39**, dacă **n=100**;
**C**. **38**, dacă **n=100**;
**D**. Numărul de pătrate perfecte egale cu $\mathbf{p^{2k} \le n}$ , cu **p** prim şi **k≥2**.

**1.64.** Ce returnează algoritmul? (sursă: personală)

```
Algorithm Calcul(i,j, k)
            {i,j,k naturale, k>0}
   iMK← i MOD k
   r  ← 1
   While (j>0) execute
      r← (r * iMK) MOD k
      j← j-1
   EndWhile
   return r
SfAlgorithm
```

**A. r=2** pentru **i=100**, **j=100** şi **k=7**;
**B. r=1** pentru **i=1000**, **j=1000** şi **k=7**;
**C. r=1** pentru **i=n+1**, **j** oarecare şi **k=n, n>2**;
**D. r=2** pentru **i=1000**, **j=1000** şi **k=7**.

**1.65.** Ce returnează algoritmul y(**a,b**) ? (sursă: personală)

```
      {a,b naturale nenule, a≥b}
Algorithm y(a,b):
  If b = 1
     then
       return a
  EndIf
  If b > a
     then
       return 0
  EndIf
  return (a-b+1)*y(a,b-1)
EndAlgorithm
```

**A.** **1**, pentru **y(1,1);**
**B.** **3**, pentru **y(3,2);**
**C.** **20**, pentru **y(5,2);**
**D.** Combinări de **a** luate câte **b**.

**1.66.** Care algoritmi calculează corect combinări de **n** luate câte **k**? (sursă: personală)

A.
```
Algorithm a(n, k)  {n,k naturale}
   If n < k
      then
         return 0
   EndIf
   If (k = 0) OR (k = n)
      then
         return 1
   EndIf
   return (n*a(n-1,k)) DIV (n-k)
EndAlgorithm
```

B.
```
Algorithm b(n, k)  {n,k naturale}
   If n < k
      then
         return 0
   EndIf
   If k = 1
      then
         return n
   EndIf
   return (n-k+1)*b(n,k-1)
EndAlgorithm
```

C.
```
Algorithm c(n,k) {n, k naturale}
   If (n=0) sau (n=1)
      then
           return 1
   return n*c(n-1,k)
EndAlgorithm
```

D.
```
Algorithm d(n, k)     {n,k naturale}
   If n < k
      then
           return 0
   EndIf
   If (k = 0) OR (k = n)
      then
           return 1
   EndIf
   If k = 1
      then
           return n
   EndIf
   return (n*d(n-1,k-1)) DIV k
EndAlgorithm
```

**1.67.** Algoritmii 1 şi 2 afişează toate tripletele **(i, j, k)** de numere naturale care verifică condiţiile:

$i^2 + j^2 = k^2$

$3 \le i < j < k \le n$ **(n** dat**)**

(sursă: personală)

```
1.    Algorithm Pitagora1 (n):
        cont← 0                  {cont este contor pentru triplete}
        For i← 3,n-2 execute {tripletele pitagorice incep cu 3,4,5}
          For j← i+1,n-1 execute
            For k← j+1, n execute
              If i² + j² = k²
                then
                     cont← cont+1
                     Write  cont,i,j,k,newline
              EndIf
            EndFor
          EndFor
         EndFor
     SfAgoritm
```

```
2. Algorithm Pitagora2 (n):
     cont← 0                              {cont este contor pentru triplete}
     nPeRad2← n DIV √2
     For i← 3,nPeRad2 execute  {tripletele pitagorice incep cu 3,4,5}
        iLa2← i²
        For j←i+1,n-1 execute
           sumaPat← iLa2+ j²
           k←[ √sumaPat ]
           If (k ≤  n) AND (k² = sumaPat)
                then
                     cont← cont+1
                     Write cont,i,j,k,newline
           EndIf
        EndFor
     EndFor
  SfAgoritm
```

Care afirmaţii sunt adevărate?:

- **A.** Algoritmul **1** este mai eficient;
- **B.** Algoritmul **2** este mai eficient;
- **C.** Ambele metode sunt la fel de eficiente;
- **D.** Algoritmul **2** are complexitatea **$O(n^2)$**.

**1.68.** Ce returnează algoritmul următor? (sursa: personală)

```
Algorithm s(n, d):     {n, d naturale, nenule iar d=2 la apelul primar}
   If  n ≤ 1
     then
          return 0
   EndIf
   If d*d > n
     then
          return 1
   EndIf
   If d*d = n
     then
          return 1 + d
   EndIf
   If  (n MOD d) = 0
     then
          return d + (n DIV d) + s(n,d+1)
   EndIf
   return s(n,d+1)
 EndAlgorithm
```

**A.** Suma divizorilor lui **n**, (**n>1**);
**B.** Suma divizorilor proprii, ai lui **n**, (**n>1**);
**C.** Suma divizorilor lui **n** fără **n**, (**n>1**);
**D.** Suma divizorilor improprii al lui **n**, (**n>1**).

**1.69.** Care din algoritmii de calcul al $C_n^k$ (combinări de **n** luate câte **k**) sunt mai eficienţi (ca număr de autoapeluri)? (**n**, **k** naturale) (sursă: personală)

```
1. Algorithm c1 (n, k){
     If k > n
       then
         return 0
     EndIf
     If (k = 0) OR (k = n)
       then
         return 1
     EndIf
     If k = 1
       then
         return n
     EndIf
     return c1(n-1,k-1)+c1(n-1,k)
  SfAlgortim
```

```
2. Algorithm c2 (n, k){
     If k > n
       then
         return 0
     EndIf
     If (k = 0) OR (k = n)
       then
         return 1
     EndIf
     If k = 1
       then
         return n
     EndIf
     return (c2(n-1,k-1)*n) DIV k
  SfAlgortim
```

```
3. Algorithm c3 (n, k){
     If k > n
       then
         return 0
     EndIf
     If (k = 0) OR (k = n)
       then
         return 1
     EndIf
     If k = 1
       then
         return n
     EndIf
     return (c3(n-1,k)*n) DIV (n-k)
  SfAlgortim
```

**A.** Metodele **1** şi **2** mai eficiente ca **3**;
**B.** Metodele **1** şi **3** mai eficiente ca **2**;
**C.** Metodele **2** şi **3** mai eficiente ca **1**;
**D.** Toate **3** sunt la fel de eficiente.

**1.70.** Algoritmul trebuie să afişeze soluţiile probl. $\overline{abc} = a^3 + b^3 + c^3$, **a**,**b**,**c** cifre. Ce instrucţiune trebuie pusă în locul celor 3 puncte, pentru cifra **b**. (sursă: personală)

```
Algorithm sumaCub()
  For i← 100,999 execute
    c← i MOD 10
    ...
    a← i DIV 100
    If a^3 + b^3 + c^3 = i
      then
         Write i, newline
    EndIf
  EndFor
EndAlgorithm
```

**A.** `b← (i DIV 10) DIV 10`

**B.** `b← (i MOD 10) DIV 10`

**C.** `b← (i DIV 100) MOD 10`

**D.** `b← (i DIV 10) MOD 10`

1.71. Se dau 2 numere naturale **a,b**, astfel ca **1<a<b**. Algoritmul **C(a,b)** apelează **d2p(x,p,q)**, iar **d2p** apelează algoritmul **prim(k)** care returnează **1**, dacă **k** e prim şi **0** în caz contrar. Algoritmul **prim** se consideră ştiut. (sursa: personală)

```
Algorithm C(a,b)
  x← a
  While  x < b                              {p1,p2,p3,p4 sunt determinati în d2p}
    If (d2p(x,p1,p2) = 1) AND  (d2p(x+1,p3,p4) = 1)
       then
           Write x,"=",p1,"*",p2
           Write x+1,"=",p3,"*",p4
    EndIf
    x← x+1
    EndIf
EndAlgorithm

Algorithm d2p(x,p,q)          {p,q transmisi prin referinţă !!!}
   d← 2
   While  d*d ≤ x execute
     If x = d * (x DIV d)
       then
          If (prim(d)=1) AND  (prim(x DIV d)=1)
             then
               p← d
               q← x DIV d
               return 1
          EndIf
     SfDaca
     If d = 2
       then
         d← d+1
       else
         d← d+2
     EndIf
  EndWhile
  return 0;
EndAlgorithm
```

Ce se afişează?

**A. Dacă a=115** şi **b=125**

**118=2*59 119=7*17**
**121=11*11 122=2*61**
**122=2*61 124=2*62**

**B. Dacă a=115** şi **b=125**

**118=2*59 119=7*17**
**121=11*11 122=2*61**
**122=2*61 123=3*41**

**C. Dacă a=30** şi **b=40**

**33=3*11 34=2*17**
**34=2*17 35=5*7**
**37=1*37 38=2*19**

**D. Dacă a=30** şi **b=40**

**33=3*11 34=2*17**
**34=2*17 35=5*7**
**38=2*19 39=3*13**

**1.72.** Algoritmul următoR, afişează triplete „heronice”. Un triplet **(i,j,k)** de numere naturale se numeşte „**heronic**”, dacă îndeplineşte condiţiile:

1) **i,j,k** sunt măsurile laturilor unui triunghi, **i ≤ j < k**.
2) aria triunghiului se exprimă printr-un număr întreg de unităţi. (sursa: personală)

```
Algorithm heron(n):
   cont← 0
   For i← 3,n-1 execute
     For j← i,n-1 execute
       For k← j+1,n execute
          If i > k-j
            then
              p  ← (i+j+k) DIV 2
              ap ← p*(p-i)*(p-j)*(p-k)
              rad← [ √ap ]
              If rad * rad = ap
                 then
                   cont← cont+1
                     Write cont,". ",i,j,k," Aria=", rad, newline
               EndIf
           EndIf
       EndFor
     EndFor
   EndFor
EndAlgorithm
```

Care afirmaţii sunt adevărate?

**A.** dacă **n=10** atunci se afişează:

**1.** **3 4 5** Aria= **6**
**2.** **5 5 6** Aria= **12**
**3.** **5 5 7** Aria= **13**
**4.** **5 5 8** Aria= **12**

**B.** dacă **n=10** atunci se afişează:

**1.** **3 4 5** Aria= **6**
**2.** **5 5 6** Aria= **12**
**3.** **5 5 8** Aria= **12**
**4.** **5 5 10** Aria= **20**

**C.** dacă **n=10** atunci se afişează:

**1.** **3 4 5** Aria= **6**
**2.** **5 5 6** Aria= **12**
**3.** **5 5 8** Aria= **12**
**4.** **6 8 10** Aria= **24**

**D.** Toate tripletele pitagorice sunt şi „**heronice**”.

**1.73.** Ce afişează următoarea secvenţă de instrucţiuni în pseudocod? (sursa: personală)

```
Read n {număr natural}
k ← 1
While n ≥ 1 execute
   If n > k
     then i ← k
     else i ← n
   EndIf
   n ← n-i
   While i ≥ 1 execute
     Write k,","
     i ← i-1
   EndWhile
   k ← k+1
EndWhile
```

**A. n=9 ⇒ 1, 2, 2, 3, 3, 4, 4, 5, 5;**
**B. n=10 ⇒ 1, 1, 2, 2, 3, 3, 4, 4, 5, 5;**
**C. n=11 ⇒ 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2;**
**D. n=12 ⇒ 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5.**

**1.74.** Se dă o matrice **A** pătratică de ordinul **n**, **n>1**. Matricea este booleană (conţine doar valori de **0**,**1**). Ştiind că numărul de valori **1** de este două treimi din numărul de valori **0**, aflaţi ce valorea poate lua **n**? (sursa: personală)

**A. n= 4;**
**B. n=10;**
**C. n=12;**
**D. n=15.**

**1.75.** O matrice **A** pătratică de ordinul **n**,(**n>1**), este booleană (conţine doar valori de **0**,**1**) şi simetrică faţă de prima diagonală. Indicaţi numărul de matrice care se pot construi: (sursa: personală)

**A. n=2 ⇒ 512** matrice;
**B. n=3 ⇒ 81** matrice;
**C. n=4 ⇒ 1024** matrice;
**D. n=5 ⇒ 625** matrice.

**1.76.** Cei **n** participanţi la o competiţie sunt organizaţi în **m** echipe, fiecare participant făcând parte dintr-o singură echipă. În timpul competiţiei fiecare participant dintr-o echipă devine prieten cu fiecare din ceilalţi participanţi din aceeaşi echipă. Dându-se **m** şi **n** aflaţi numărul maxim şi numărul minim de prietenii potenţiale. Care afirmaţii sunt adevărate? (sursa: folclorul matematic)
Exemplu: **n = 6** participanţi şi **m = 3** echipe ⇒ **Max=6** pentru distribuţia **1,1,4**
iar **Min=3** pentru distribuţia **2,2,2**.

**A. n =11, m = 3 ⇒ Max=36** iar **Min=15**
**B. n =12, m = 4 ⇒ Max=36** iar **Min=10**
**C. n =50, m =20 ⇒ Max=190** iar **Min=30**
**D. n =15, m = 6 ⇒ Max=45** iar **Min=12**

**1.77.** Fie secvenţa: (sursa: personală)

```
Read m,n,x {numere natural nenule, m<n}
p←0
While (m < n) AND (p = 0) execute
  If (m MOD x = 0) AND (n MOD x = 0)
     then
          p← x
     else
       If m MOD x = 0
          then
               n← n-1
          else
               m← m+1
      EndIf
  EndIf
EndWhile
Write  m," ",n
```

Care afirmaţie e adevărată?

**A. m=30, n=40, x=17,** se afişează **18 19**
**B. m=10, n=40, x=17,** se afişează **18 20**
**C. m=11, n=30, x=7,** se afişează **14 28**
**D. m=14, n=15, x=5,** se afişează **15 15**

**1.78.** Avem **9** monede de aceeaşi fomă şi culoare. **8** monede au aceeaşi masă, iar una are cu **1** gram mai puţin (nu ştim care monedă). Dispunem de o balanţă (cântar) cu **2** talere. Pe fiecare taler al balanţei se pot pune oricâte monede. Care este numărul minim de utilizări ale balanţei (indiferent de aranjarea monedelor) pentru a afla care este moneda cu masa mai mică. (sursa: folclor matematic)

**A. 4**
**B. 1**
**C. 2**
**D. 3**

**1.79.** Un individ (mai tânăr) trebuie să urce o scară de **n** trepte (în drumul spre casă). Treptele le poate urca câte una sau câte două (fiind mai sportiv). În câte moduri poate urca cele n trepte? (sursa: personală)

**A. n=7 => 12 moduri**
**B. n=8 => 34 moduri**
**C. n=10 => 53 moduri**
**D. n=11 => 144 moduri**

**1.80.** O comunitate de oameni este împărţită în **3** grupuri etnice. Câte legături duale se pot forma între cele **3** grupuri etnice (legătură dual = un membru dintr-un grup cu un membru din alt grup)? (sursa: personală)

**A.** Primul grup:**3** membri, al **2**-lea grup**: 4** şi ultimul grup are **2 => 25** legături;
**B.** Primul grup:**3** membri, al **2**-lea grup: **5** şi ultimul grup are **2 => 28** legături;
**C.** Primul grup:**3** membri, al **2**-lea grup: **4** şi ultimul grup are **2 => 26** legături;
**D.** Primul grup:**3** membri, al **2**-lea grup: **5** şi ultimul grup are **2 => 31** legături.

**1.81.** Ce returnează algoritmul **cc(n,k)** pentru **n** şi **k** numere naturale ca date de intrare. (sursa: personală)

```
Algorithm cc(n, k):
   c← 1
  For  i← 1,k execute
       c← (c*(n-i+1)) DIV i
   EndFor
   returnează c;
EndAlgorithm
```

**A. n=15, k=10 ⇒ 3001;**
**B. n=15, k=10 ⇒ 3003;**
**C. n=19, k=15 ⇒ 3877;**
**D. n=19, k=15 ⇒ 3876;**

**1.82.** Ce afişează algoritmul **p(a)**, **a** număr natural nenul. (sursa: personală)

```
Algorithm p(a):
    nC← [log10(a)]
    p ← 10^nC
    b ← a
    Repeat
       b←  (b MOD 10)*p + b DIV 10
       Write b
    Until (a = b)
EndAlgorithm
```

**A. a=1234 ⇒ 2341 3412 4123 1234**
**B. a=1234 ⇒ 4123 1234 2341 3412**
**C. a=1234 ⇒ 4123 1234 3412 2341**
**D. a=1234 ⇒ 4123 3412 2341 1234**

**1.83.** Alegeţi afişările corecte al algoritmului sp(n) pentru n întreg? (sursa: personală)

```
Algorithm sp(n):
  For f← 2,n execute
    p← 0
    While n MOD f=0 execute
       n← n DIV f
       p← p + 1
    EndWhile
    Write p
 EndFor
EndAlgorithm
```

**A. n= -10,** afişează **11**
**B. n= 10,** afişează **1001**
**C. n= 100,** afişează **2002**
**D. n= 121,** afişează **0000000003**

**1.84.** Ce returnează algoritmul? (sursa: personală)

```
Algorithm  sp(n,p):
   If  p > n
     then
       Returnează  0
     else
       return (n DIV p)+ sp(n,p+p)
   EndIf
EndAlgorithm
```

**A.** pentru **n=10, p=2,** returnează **8;**
**B.** pentru **n=11, p=2,** returnează **8;**
**C.** pentru **n=12, p=2,** returnează **8;**
**D.** pentru **n=12, p=3,** returnează **7.**

**1.85.** De câte ori se execută instrucţiunea de la linia **6** pentru **n**=**798659**? (sursa: personală)

```
1.Algorithm f(n):
2.   If (n<10)
3.       then
4.                return n
5.   EndIf
6.   u← n MOD 10
7.   p← f(n DIV 10)
8.   If u > p
9.       then
10.               return u
11.      else
12.               return p
13.  EndIf
14.EndAlgorithm
```

**A**. de **6** ori;
**B**. de **5** ori;
**C**. de acelaşi număr de ori ca pentru **n**=**123456**
**D**. de acelaşi număr de ori ca pentru **n**=**7654321**.

**1.86.** De câte ori se execute instrucţiunea de la linia **12** (**returnează p**) pentru **n=798659**? (sursa: personală)

```
1.Algorithm f(n):
2.  If (n<10)
3.      then
4.                return n
5.  EndIf
6.  u← n MOD 10
7.  p← f(n DIV 10)
8.  If u < p
9.      then
10.               return u
11.     else
12.               return p
13. EndIf
14.EndAlgorithm
```

**A.** de **4** ori;
**B.** de **3** ori;
**C.** de acelaşi număr de ori ca pentru **n=76543**;
**D.** de acelaşi număr de ori ca pentru **n=123456**.

**1.87.** Care afirmaţii sunt adevarate? (sursa: personală)

```
          {nr=1, la apelul primar}
1  Algorithm a(nr):
2    Read x
3    If (x=0)
4       then
5               return 0
6    EndIf
7    If (x MOD 3 = 0)
8       then
9               return (x MOD 10)+ a(nr+1)
10      else
11              return  a(nr+1)
12   EndIf
13 EndAlgorithm
```

**A.** Pt. şirul **x**: **1, 2, 3, 4, 5, 6, 0** returnează **10**;
**B.** Pt. şirul **x: 15, 24, 25, 25, 25, 222, 0** returnează **11**;
**C.** Pt. şirul **x: 1, 2, 3, 4, 5, 6, 0** instrucţiunea de la linia **11** se execute de $2^2$ ori;
**D.** Pt. şirul **x: 1, 2, 3, 4, 5, 6, 0** returnează **9**.

**1.88.** De câte ori se execute instrucţiunea de la linia **12** pentru **n=798659**? (sursa: personală)

```
1.Algorithm f(n):
2.  If (n<10)
3.    then
4.         return n
5.  EndIf
6.  u← n MOD 10
7.  p← f(n DIV 10)
8.  If u < p
9.      then
10.              return u
11.     else
12.              return p
13. EndIf
13.EndAlgorithm
```

**A**. de **4** ori;
**B**. de **3** ori;
**C**. de acelaşi număr de ori ca pentru **n=3456**;
**D**. de acelaşi număr de ori ca pentru **n=12345**.

**1.89.** Care afirmaţii sunt false? (sursa: personală)

```
1 Algorithm p(n,d):
2   If n ≤ 1
3      then
4                return false
5   EndIf
6   If(n > 2) AND (n MOD 2=0)
7       then
8                return false
9   EndIf
10  If d*d > n
11    then
12               return true
13  EndIf
14  If n MOD d = 0
15      then
16               return false
17      else
18               return p(n,d+2)
19  EndIf
20 EndAlgorithm
```

**A.** la apelul **p(10,3)** comparaţia (If-ul) de pe linia **6** se execute de **2** ori;

**B.** la apelul **p(27,3)** instrucţiunea de pe linia **18** (returnează **p(n,d+2)**) se execute de **2** ori;

**C.** la apelul **p(29,3)** instrucţiunea de pe linia **18** (returnează **p(n,d+2)**) se execute de **2** ori;

**D.** la apelul **p(11,2)** comparaţia (If-ul) de pe linia **6** se execute **3** ori.

**1.90.** Fie algoritmul **calcul(a,b)** cu parametrii **a**,**b** numere naturale, **1 ≤ a,b ≤ 1000**. (Din concurs mai vechi)

```
Algorithm calcul (a,b)
  If a ≠ 0
    then
         return Calcul(a DIV 2, b+b) + b*(a MOD 2)
    else
         return 0
  EndIf
EndAlgorithm
```

Care afirmaţii sunt adevărate?

- **A.** Dacă **a** şi **b** sunt egale, algoritmul returnează valoarea lui **a**;
- **B.** Dacă **a=100** şi **b=2**, algoritmul se autoapeleză de **7** ori;
- **C.** Valoarea calculată şi returnată de algoritm este **a+a+...+a** de **b** ori;
- **D.** Algoritmul returnează **a*b**.

**1.91.** Ce se va afișa pe ecran în urma executerii următoarelor instrucțiuni? (sursa: Politehnica Bucureşti)

```
For i← 1,6 execute
   For j← 6,i,-1 exceută
     If (i MOD 2) = 0
       then
            Write i
       else
            Write j
     EndIf
   EndFor
 EndFor
```

**A.** 654321222226543444456
**B.** 654321222226543444556
**C.** 654321222226543444666
**D.** 654321222226543444656

**1.92.** Pe o masă este scrisă ecuaţia **a + b = c** cu cifre de piese de rummy (cele de la 1 la 9 indiferent de culoare). Dupa un cutremur puternic, s-au permutat toate cifrele şi semnele matematice între ele şi s-a obţinut o noua "**ecuaţie**" (evident, greşită). (sursă: model FMI U.Bucureşti)

```
129129851 = 29552 + 1177003
```

Care ar fi putut fi valoarea iniţială a lui c?

**A.** 14010927
**B.** 16038950
**C.** 15111922
**D.** 18839920

**1.93.** Ce afişează următorul algoritm (care sunt corecte): (sursa: personală)

```
Algorithm t(n):
   For i← 1,(n DIV 2) execute
     s← i
     j← i+1;
     While s < n execute
       s← s + j
       j← j + 1
     EndWhile
     If s = n
       then
             Write n,"="
             For k← i,j-1,1
                  Write k,"+"
             EndFor
             {se şterge ultimul "+"}
             Write newline
     EndIf
   EndFor
EndAlgorithm
```

```
A. pentru n=15 =>    15=1+2+3+4+5
                     15=7+8

B. pentru n=9  =>    9=2+3+4

C. pentru n=21 =>    21=1+2+3+4+5+6
                     21=6+7+8
                     21=10+11

D. pentru n=30 =>    30=4+5+6+7+8
                     30=9+10+11
```

**1.94.** Ce afişează următorul algoritm (care sunt corecte): (sursa: personală)

```
Algoritmul t(n):
   For i← 1,[√n] execute
     s← i
     j← i;
    While s < n execute
        j← j + 1
        s← s * j
     EndWhile
     If s = n
       then
             Write n,"="
             For k← i,j-1,1
                  Write k,"*"
             EndFor
             {se şterge ultimul "*"}
              Write newline
     EndIf
   EndFor
EndAlgorithm
```

**A.** pentru n=24 =>
24=1*2*3*4*5
24=1*2*3*4

**B.** pentru n=120 =>
120=1*2*3*4*5
120= 2*3*4*5

**C.** pentru n=720 =>
720=1*2*3*4*5*6
720=2*3*4*5*6
720=8*9*10

**D.** pentru n=72 => 72=8*9

**1.95.** Se întâlnesc doi prieteni, să zicem **X** şi **Y**, după mai mulţi ani. (sursa: foclor matematic)

**X** - ce mai faci, ai copii?
**Y** – m-am căsătorit şi am **3** copii de când nu ne-am mai văzut.
**X** – ce vârstă au?
**Y** – te ştiu matematician/informatician, îţi dau **2** repere:
1) Produsul vârstelor lor este **36**.
2) Iar suma vârstelor lor este egală cu numărul de maşini din parcarea lângă care ne aflăm.

**X** gândeşte... ... ... şi după un timp spune:
– mai am nevoie de un reper.
**Y** – cel mai mare băiat al meu are părul blond.
**X** – Ok şi îi spune vârstele celor **3** copii.

Care afirmaţii sunt adevărate?

**A.** Copilul cel mare are varsta de **12** ani;
**B.** Cei **3** copii au vârstele: **1,4,9** ani;
**C.** Cel mai mare copil are vârsta de **9** ani;
**D.** Numărul maşinilor din parcare este **13**.

**1.96.** Se dă numărul **32767** în baza **10**. Care afirmaţii sunt adevărate? (sursa: personală)

**A.** $32767 = 111111111111110_{(2)}$
**B.** $32767 = 111111111111111_{(2)}$
**C.** $32767 = 77777_{(8)}$
**D.** $32767 = 7FFF_{(16)}$

**1.97.** Se dă numărul **1024** în baza **10**. Care afirmaţii sunt adevărate? (sursa: personală)

**A.** $1024 = 1101221_{(3)}$
**B.** $1024 = 1101220_{(3)}$
**C.** $1024 = 1358_{(9)}$
**D.** $1024 = 1357_{(9)}$

**1.98.** Trei inşi **A**,**B**,**C** sunt aşezaţi în coloană, **C** este primul, **B** al-**2**-lea şi apoi **A**. Există **5** pălării de **2** culori (**3** roşii şi **2** albe). Li se pune pe cap câte o pălarie, fără să cunoască culoarea (random). **A** (a lui nu şi-o vede), vede palaria lui **B** şi pălaria lui **C. B** (a lui nu şi-o vede) vede doar pălăria lui **C**) iar **C** nu vede nici o pălărie (nici a lui). (sursa: folclor matematic)
Este întrebat **A** ce culoare are pălaria sa;
A- Nu ştiu
Este întrebat apoi **B** ce culoare are pălaria sa;
B- Nu ştiu
Este întrebat şi **C**, ce culoare are pălăria sa;
C – eu ştiu

Care afirmaţii sunt adevărate:

**A.** **C** are pălărie de culoare albă;
**B.** **B** are pălarie de culoare roşie;
**C.** **A** şi **B** au pălării de culori diferite;
**D.** **C** are pălărie de culoare roşie.

**1.99.** La concursul de admitere la **FMI**, se dau teste-grile. La fiecare grilă se poate da **1** răspuns sau **2** răspunsuri sau **3** răspunsuri din cele **4** întrebări.
Câte răspunsuri sunt posibile la o grilă. (sursa: personală)

**A.** **4! / 2 + 2** răspunsuri;
**B.** **12** răspunsuri;
**C.** **2*4 +2*3** răspunsuri;
**D.** **13** răspunsuri.

**1.100.** Un test alcatuit din **30** de intrebari a fost notat astfel: pentru fiecare răspuns corect s-au dat **9** puncte şi pentru fiecare răspuns greşit s-au scăzut **6** puncte. Un candidat a obţinut în final **0**(zero) puncte.
Care propoziţii sunt adevărate?

**A.** Candidatul a dat **8** răpunsuri corecte;
**B.** Candidatul a dat **12** răspunsuri corecte;
**C.** Candidatul a dat **22** răspunsuri greşite;
**D.** Candidatul a dat **18** răspunuri greşite.

**1.101.** Un triunghi echilateral şi un hexagon regulat au perimetrele egale. Care afirmaţii sunt adevărate?

- **A.** Triunghiul şi hexagonul au arii egale;
- **B.** Triunghiul are aria mai mare;
- **C.** Raportul ariilor A(triunghi)/A(hexagon) **= 2/4;**
- **D.** Raportul ariilor A(triunghi)/A(hexagon) **= 2/3.**

**1.102.** Fie schema de „calcul”: (sursă: folclorul matematic)

**1 ⊕ 4 = 5**
**2 ⊕ 5 = 12**
**3 ⊕ 6 = 21**
**4 ⊕ 7 = 32**

Ce valoare va avea **6 ⊕ 9 ?**

- **A. 32**
- **B. 45**
- **C. 60**
- **D. 77**

**1.103.** Se consideră următorul subalgoritm: (sursa: admitere/concurs FMI UBB Cluj-Napoca)**.**

```
Algorithm s(a):
      If a < 50
        then
            If a MOD 3 = 0
                  then
                        return s(2 * a - 3)
                  else
                        return s(2 * a - 1)
            EndIf
        else
            return a
      EndIf
SfAlgorithm
```

For care dintre valorile parametrului de intrare **a** subalgoritmul va returna valoarea **61**?

**A. 16**
**B. 61**
**C. 4**
**D. 31**

**1.104**. Care dintre algoritmii care urmează returnează cel mai mare multiplu al numărului natural ***a***, multiplu care este mai mic sau egal cu numărul natural ***b*** (0 < ***a*** < 10 000, 0 < ***b*** < 10 000, ***a*** < ***b***)?
(sursa: admitere/concurs FMI UBB Cluj-Napoca)

**A.**
```
Algorithm f(a, b):
          c ← b
          While c MOD a = 0 execute
             c ← c - 1
          EndWhile
          returnează c
       EndAlgorithm
```

**B.**
```
Algorithm f(a, b):
       If a < b
         then
             return f(2 * a, b)
         else
             If a = b
                  then
                       return a
                  else
                       return b
           EndIf
        EndIf
      SfAlgorithm
```

**C.**
```
Algorithm f(a, b):
            return (b DIV a) * a
     EndAlgorithm
```

**D.**
```
Algoritmul f(a, b):
         If b MOD a = 0
            then
                    return b
         EndIf
         return f(a, b - 1)
      SfAlgorithm
```

**1.105.** Ce returnează subprogramul **f(n)** pentru **n** număr natural? (sursă: personală)

```
Algorithm f(n):
  If n<10
    then
      If n MOD 2 > 0
           then
                return n
           else
                return -1
      EndIf
  EndIf
  If n MOD 2 > 0
    then
           u← n MOD 10
    else
           u← -1
  EndIf
  p← f(n DIV 10)
  If u > p
     then
           return u
     else
           return p
  EndIf
EndAlgorithm
```

**A**. **n**, pentru **n=0**;
**B**. **-1**, pentru **n=8468**;
**C**. **5**, pentru **n=98765**;
**D**. **7**, pentru **n=73137**.

**1.106.** Care secvenţe returnează lungimea sufixului celui mai lung (a lui **n**) care se repetă?
(**n** are cel mult **20** de cifre, **n** natural, vectorul c are 21 de componente).

```
A. Algorithm sufixA(n):   {n=83456345  => lung =3 sau n=134343    => lung =3}
      nn     ← n
      nrCifre← 0
      While(nn>0) execute
            nrCifre← nrCifre + 1
            nn     ← nn DIV 10
      EndWhile
      nr← nrCifre
      While n>0 execute
            c[nr]← n MOD 10
            nr   ← nr-1
            n    ← n DIV 10
      EndWhile
      lung← 0                        ///determinare lungime sufix
      For i← nrCifre,i>1,-1 Execute
            k← 0
            While (nrCifre-k>1) AND (i-k>1) AND (c[n-k]= c[i-k-1]) execute
                  k← k+1
            SfCătTimp
            If k>lung
               then
                  lung← k
            EndIf
      EndFor
      return lung
   EndAlgorithm

B. Algorithm sufixB(n):
      nn      ← n
      nrCifre← 0
      While(nn>0) execute
            nrCifre← nrCifre + 1
            nn     ← nn DIV 10
      EndWhile
      nr← nrCifre
      While n>0 execute
            c[nr]← n MOD 10
            nr   ← nr-1
            n    ← n DIV 10
      EndWhile
      lung← 0                        ///determinare lungime sufix
      For i← nrCifre,i>1,-1 Execute
         k← -1
         While (nrCifre-k>1) AND (i-k>1) AND (c[nrCifre-k]= c[i-k-1]) execute
            k← k+1
         SfCătTimp
         If k>lung
           then
                  lung← k
        EndIf
      EndFor
      return lung
   EndAlgorithm
```

```
C. Algorithm sufixC(n):
      nn      ← n
      nrCifre← 0
      While(nn>0) execute
            nrCifre← nrCifre + 1
            nn     ← nn DIV 10
      EndWhile
      nr← nrCifre
      While n>0 execute
            c[nr]← n MOD 10
            nr   ← nr-1
            n    ← n DIV 10
      EndWhile
      lung← 0                        ///determinare lungime sufix
      For i← nrCifre,i>1,-1 Execute
         k← 0
         While (nrCifre-k>1) AND (i-k>1) AND (c[nrCifre-k]= c[i-k-1]) execute
                  k← k+1
         SfCătTimp
         If k < lung
           then
                  lung← k
        EndIf
      EndFor
      return lung
   EndAlgorithm

D. Algorithm sufixD(n):
      nn      ← n
      nrCifre ← 0
      While(nn>0) execute
        nrCifre← nrCifre + 1
        nn     ← nn DIV 10
      EndWhile
      nr← nrCifre
      While n>0 execute
        c[nr]← n MOD 10
        nr   ← nr-1
        n    ← n DIV 10
      EndWhile
      lung← 0                        ///determinare lungime sufix
      For i← nrCifre,i>1,-1 Execute
         k← 0
         While (nrCifre-k>1) AND (i-k>1) AND (c[nrCifre-k]= c[i-k-1]) execute
                  k← k+1
         SfCătTimp
         If k > lung
           then
                  lung← k
        EndIf
      EndFor
      return lung
  EndAlgorithm
```

**1.107.** Se dă următoarea porţiune de pseudocod . Care afirmaţii sunt false?

```
Read n
For i← 1,n execute
   j← i*i
   Write j
EndFor
```

**A.** După instrucţiunea **For**, **j** are valoarea $(n-1)^2$;
**B.** După instrucţiunea **For**, **j** are valoarea **n**;
**C.** După instrucţiunea **For**, **j** are valoarea $(n+1)^2$;
**D.** Toate din afirmaţiile **A.**, **B.**, **C.** sunt false.

**1.108.** Care algoritmi calculează corect combinări de **n** luate câte **k,** pentru orice valori naturale ale lui **n** şi **k**. (sursa: personală)

```
A.Algortihm A(n,k):  {n,k naturale}
   If n < k
      then
           return 0
   EndIf
   If (k=0) OR (n=k)
      then
           return  1
   EndIf
   n← n-1
   k← k-1
   return A(n,k+1)+A(n,k)
EndAlgorithm
```

```
B.Algorithm B(n,k)   {n,k>1 naturale}
   If n < k
      then
           return 0
   EndIf
   If k = 1
      then
           return n
   EndIf
   return ((n-k+1) DIV k)*B(n,k-1)
EndAlgorithm
```

```
C. Algorithm C(n,k):   {n,k naturale}
   If (n=0) OR (n=1)
      then
           return 1
   EndIf
   n← n-1
   return (n+1)*C(n,k)
EndAlgorithm
```

```
D. Algorithm D(n, k):    {n,k naturale}
   If n < k
      then
           return 0
   EndIf
   If k = 1
      then
           return n
   EndIf
   If (k=0) OR (k=n)
      then
           return 1
   EndIf
   return  (D(n-1,k-1) DIV k)*n
EndAlgorithm
```

**1.109.** La caseria Operei de Stat se vând bilete pentru un spectacol. Se aşează la rând **n** cumpărători de bilet. Biletul de spectacol costă 25 de lei. Fiecare cumpărător poate să aibă bancnote de 25, 50, 100 de lei. (bancnotele care circulă în ţară). Există cumpărători care nu au toate tipurile de bancnote**.** Mai există resctricţia că fiecare cumpărător poate cumpăra doar un bilet. În momentul când nu se poate da rest se opreşte vânzarea de bilete. Următorul algoritm simulează cumpărarea de bilete. La început caseria nu are nici o bancnotă. Care afirmaţii sunt adevărate? {sursă personală}.

```
Algorithm bilete(n):      {n>0, şi reprezintă numărul de cumparatori ce stau la rand}
   c25← c50← c100← 0
   For i← 1,n execute
     Citeste val  {caseria primeşte bancnota
     If val = 25
       then c25← c25+1
              {se vinde biletul}
       else
         If val = 50
           then
             If c25 > 0
               then
                 c25← c25-1
                 c50← c50+1
                    {se vinde biletul}
               else
                 return false
             EndIf
           else  {val este 100 sigur}
             If (c50 > 0) AND (c25 > 0)
               then
                 c25← c25-1
                 c50← c50-1
                 c100←c100+1
                   {se vinde biletul}
               else
                 If c25 ≥ 3
                   then
                     c25← c25-3
                     c100←c100+1
                      {se vinde biletul}
                   else
                     return false
                 EndIf
            EndIf
         EndIf
     EndIf
   EndFor
   return true
 EndAlgorithm
```

A. dacă n=5 şi şirul bancnotelor cumpărătorilor este: 25,50,25,100,100 se returnează true;
B. dacă n=5 şirul bancnotelor cumparatorilor este: 25,25,50,100,100 atunci caseria a vândut 5 bilete
C. dacă n=10 şirul bancnotelor cumpărătorilor este: 25,50,25,25,50,50,100,25,50,50 se returnează true şi caseria are:
   1 bancnota de 100
   3 bancnote 50
   3 bancote de 25

D. dacă n=10 şirul bancnotelor cumpărătorilor este: 25,50,25,50,25,50,25,25,100,50 se returnează true şi caseria are:
   1 bancnota de 100
   3 bancnote 50

**1.110.** La caseria Operei de Stat se vând bilete pentru un spectacol. Se aşează la rând **n** cumpărători de bilet. Biletul de spectacol costă 25 de lei. Fiecare cumpărător poate să aibă bancnote de 25, 50, 100 de lei. (bancnotele care circulă în ţară). Există cumpărători care nu au toate tipurile de bancnote**.** Fiecare cumparator poate cumpăra mai multe bilete. În momentul când nu se poate da rest se opreşte vânzarea de bilete. La început caseria nu are nici o bancnotă. Pentru fiecare cumpărător se dau numărul de bilete şi bancnotele de 25, 50, 100 de lei oferite casierului.

Exemple:

-(4, 0, 2, 0) înseamnă: cumpărătorul vrea să cumpere 4 bilete şi oferă 2 bancnote de 50 de lei

-(3, 0, 0, 1) înseamnă: cumpărătorul vrea să cumpere 3 bilete şi oferă 1 bancnotă de 100 de lei

Cumpărătorul oferă bancnote astfel încât suma oferită ≥ costul total de bilete, dar cât mai apropiată de acest cost.

De exemplu (3,0,0,2) e ilogică (oferă 200 de lei pentru costul a 3 bilete cu totalul de 75 de lei) . Normal ar oferi doar 100 de lei. {sursă: personală}.

Care afirmaţii sunt adevărate?

**A. Dacă n=5 cu şirul de cumpărători (2, 2,0,0) (3, 0,2,0) (2, 0,0,1) (2, 0,1,0), (5, 0,0,2) atunci caseria ii poate servi pe toţi cumpărătorii.**

**B. Dacă n=5 cu şirul de cumpărători (2, 2,0,0) (3, 0,2,0) (2, 0,0,1) (2, 0,1,0), (5, 0,0,2) atunci caseria ii poate servi pe toţi cumpărătorii şi are în casă 300 de lei.**

**C. Dacă n=5 cu şirul de cumpărători (2, 2,0,0) (3, 0,2,0) (2, 0,0,1) (2, 0, 1,0), (5, 0,0,2) atunci caseria ii poate servi pe toţi cumpărătorii şi are în casă 350 de lei sub forma: 1 bancnota de 50 şi 3 bancnote de 100 lei.**

**D. Dacă n=5 cu şirul de cumpărători (2, 2,0,0) (3, 0,2,0) (2, 0,0,1) (3, 0,0,1 ), (5, 0,1,1) atunci caseria ii poate servi pe toţi cumpărătorii şi are în casă 375 de lei.**

**1.111.** Se consideră secvenţa Pseudocod. Care valori **NU** se afişează? (sursă: personală)

```
            {n,i sunt întregi}
n← 50
i← 0
While (i<n) execute
      i← i+1
EndWhile
Write i
```

**A.** a lui **n;**
**B. 49;**
**C. 50;**
**D. 51.**

**1.112.** Ce valoare **NU** se afişează după execuţia secvenţei? (Pseudocod; sursă: personală)

```
                {n,i sunt întregi}
n← 100
For  i← 3,n,5 execute
          {corpul ciclului,  nu se modifică i}
EndFor
Write i
```

**A.** **n+3;**
**B.** **98;**
**C.** **103;**
**D.** **n-2**.

**1.113.** Care afirmaţii sunt **false**, pentru **n** întreg? (Pseudocod; sursă:personală)

```
For i←0,9 execute
      v[i]← 0
EndFor
While n>0 execute
      v[n MOD 10]←1
      n← n DIV 10
EndWhile
```

**A.** Pentru **n=139912** ⇒ **v= [0, 1, 2, 1, 0, 0, 0, 0, 0, 1];**
**B.** Pentru **n=13913** ⇒ **v= [0, 1, 0, 1, 0, 0, 0, 0, 0, 1];**
**C.** Pentru **n=-129121** ⇒ **v= [0, 1, 1, 0, 0, 0, 0, 0, 0, 1];**
**D.** Pentru **n=-17972,** ⇒ **v= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0].**

**1.114.** Care afirmaţii sunt **false**, pentru **n** întreg? (Pseudocod; sursă: personală)

```
For i←0,9 execute
      v[i]← 0
EndFor
While n>0 execute
      v[n MOD 10] ← v[n MOD 10]+1
      n           ← n DIV 10
EndWhile
```

**A.** Pentru **n=179712**, vectorul **v** are valorile **[0, 1, 1, 0, 0, 0, 0, 1, 0, 1];**
**B.** Pentru **n=12912**, vectorul **v** are valorile **[0, 1, 1, 0, 0, 0, 0, 0, 0, 1];**
**C.** Pentru **n=132712**, vectorul **v** are valorile **[0, 2, 2, 1, 0, 0, 0, 1, 0, 0];**
**D.** Pentru **n=-132912**, vectorul **v** are valorile **[0, 0, 0, 0, 0, 0, 0, 0, 0, 0].**

**1.115.** Care afirmaţii sunt **false** după execuţia algoritmului **cifre(n, m)**? (**n** întreg, **m** cifră)
(Pseudocod; sursă: personală)

```
Algorithm cifre(n,m):
  For i← 0,9 execute
    v[i]=← 0
  EndFor
  If n<0
     then
          n← -n
  EndIf
  If n=0
    then
          v[0]← 1
    else
     While n>0 execute
          u← n MOD 10
          If u MOD m = 0
               then
                    v[u]← 1
          EndIf
          n← n DIV 10
     EndWhile
  EndIf
EndAlgorithm
```

**A.** pentru **n= 1234**, **m=3** => **v = [0, 1, 1, 0, 1, 0, 0, 0, 0, 0];**
**B.** pentru **n= 9876**, **m=3** => **v = [0, 0, 0, 0, 0, 0, 1, 0, 0, 1];**
**C.** pentru **n= ⁻123456**, **m=2** => **v = [0, 0, -1, 0, -1, 0, -1, 0, 0, 0];**
**D.** pentru **n= ⁻ 9876543**, **m=5** => **v = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0].**

**1.116.** Algoritmul **a(n)**, n natural, returnează valori naturale. Care afirmaţii sunt **false**?
(Pseudocod; sursă: personală).

```
Algorithm a(n):
  If n=0
     then
          return 0
  EndIf
  For i← 0,9 execute
     v[i]← 0
  EndFor
  While n>0 execute
        v[n MOD 10] ← v[n MOD 10]+1
        n           ← n DIV 10
  EndWhile
  c← 1
  While (c < 10) AND (v[c] = 0) execute
        c← c+1
  EndWhile
  m   ← c
  v[c]← v[c]-1
  For c←0,9 execute
    While v[c]>0 execute
          m   ← m*10+c
          v[c]← v[c]-1
    EndWhile
  EndFor
  return m
EndAlgorithm
```

**A**. pentru **n=13507** se returnează **30157;**
**B**. pentru **n=40123** se returnează **10234**;
**C**. pentru **n=1203** se returnează **2013;**
**D**. pentru **n=405402** se returnează **200445**.

**1.117.** Care afirmaţii sunt **adevărate**, pentru **n** întreg? Algoritmul **a** are ca parametru un număr întreg **n**. **Din algoritmul** a se apelează algoritmul **b(v)**, unde v este un vector de 10 poziţii, numerotarea de la poziţia 0. Apoi se apelează algoritmul **c(n,v)** unde **n** este parametrul de intrare a lui **a**, iar **v** este vectorul de 10 poziţii anterior explicat.
Ce se afişează? (Pseudocod; sursă:personală)

```
Algorithm a(n):
  b(v)
  c(n,v)
  Write v
EndAlgorithm

Algorithm b(v):
  For i=0,9 execute
     v[i] ← 0
  EndFor
EndAlgorithm

Algorithm c(n,v):
   If (n>0)
     then
       v[n MOD 10]←1
       c(n DIV 10, v)
   EndIf
EndAlgorithm
```

**A.** Pentru **n=318912** ⇒ **0121000011;**
**B.** Pentru **n=13913** ⇒ **0101000001;**
**C.** Pentru **n=-4321** ⇒ **0111100000;**
**D.** Pentru **n=-17972,** ⇒ **0000000000.**

**1.118.**

Fie secvenţa următoare. Care afirmaţii sunt **false**? (sursă: personală)

```
Read n,c {n natural nenul, c cifra}
z← 0
While (n MOD 10) = c  execute
      n← n DIV 10
      z← z+1
EndWhile
Write z
```

**A.** Afişează **3**, pentru **n=123** şi **c=3**;
**B.** Afişează **2**, pentru **n=12003** şi **c=0**;
**C.** Afişează **4**, pentru **n=1277771** şi **c=7**;
**D.** Afişează **3**, pentru **n=12555** şi **c=5**.

1.119. Se consideră algoritmul afis(n), unde ***n*** este un număr întreg. (sursă: concurs/admitere FMI)

```
Algorithm afis(n):
      Write n, ", "
      If n > 0 then
            afis(n DIV 3)
            Write n, ", "
      EndIf
EndAlgorithm
```

Ce se va afișa la apelul **afis(n)**?

A. Pentru n=16 se va afişa : 16,5,1,-1,1,5,16,;
B. Pentru n=-16 se va afişa: -16,16,;
C. Pentru n=24 se va afişa: 24, 8, 2, 0, 2, 8, 24,;
D. Pentru n=27 se va afişa: 27, 9, 3, 1, 3, 9,27,.

1.120. Se consideră algoritmul afis(n), unde ***n*** este un număr întreg. (sursă: concurs/admitere FMI)

```
Algorithm afis(n):
      Write n, ", "
      If n >(n DIV 3) then
            afis(n DIV 3)
            Write n, ", "
      EndIf
EndAlgorithm
```

Ce se va afișa la apelul **afis(n)**?

A. Pentru n=27 se va afişa : 27,9,3,1,0,1,3,9,27, ;
B. Pentru n=-15 se va afişa: -15,-15, ;
C. Pentru n=-33 se va afişa: -33, ;
D. Pentru n=27 se va afişa: 27, 27,9,9,3,3,1,1,0, ;

**1.121.** Se consideră algoritmul afis(n), unde ***n*** este un număr întreg. (sursă: concurs/admitere FMI)

```
Algorithm afis(n):
      Write n, ", "
      If n > n DIV 3 then
            Write n, ", "
            afis(n DIV 3)
      EndIf
EndAlgorithm
```

Ce se va afișa la apelul **afis(n)**?

A. Pentru n=27 se va afişa : 27,9,3,1,0,1,3,9,27, ;
B. Pentru n=-16 se va afişa: -16,-16, ;
C. Pentru n=-24 se va afişa: -24, ;
D. Pentru n= 27 se va afişa: 27, 27,9,9,3,3,1,1,0, ;

**1.122.** Se consideră algoritmul cauta(n, b), unde ***n*** și ***b*** sunt numere naturale ($0 \le n \le 10^6$, $2 \le b < 10$, $c \ge 0$).
(sursă: concurs/admitere FMI)

```
Algorithm cauta(n, b, c):
      v ← 0
      If n = 0 then
            return 1
      else
            m ← n
                  While m > 0 execute
                        If m MOD b = c then
                              v ← v + 1
                        EndIf
                        m ← m DIV b
                  EndWhile
                  return v
      EndIf
EndAlgorithm
```

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

**A.** Algoritmul determină și returnează câte cifre egale cu **c** are numărul **n**.

**B.** Algoritmul determină și returnează numărul de cifre egale cu **c** din reprezentarea în baza **b** a numărului **n**, dacă c<b.

**C.** Pentru n=1456, b=4 şi c=3, algoritmul returnează 1

**D.** Pentru n=123123, b=4 şi c=3, algoritmul returnează 4.

**1.123.** Pentru a genera numerele cu ***n*** cifre formate doar din cifrele 1, 5, 8, se utilizează un algoritm care, pentru ***n*** = 2, generează în ordine descrescătoare numerele 88, 85, 81, 58, 55, 51, 18, 15, 11.

Dacă ***n*** = 4 și se utilizează același algoritm, care afirmaţii sunt **adevărate**?

**A**. Numărul total de elemente al şirului generat este **64**;

**B**. Numărul total de elemente al şirului generat este **81;**

**C**. Înaintea elementului **5555** este **5558;**

**D**. După elementul **5151** este **5115**.

**1.124.** Algoritmul ceFace(n) are ca parametru de intrare un număr natural ***n*** (0 ≤ ***n*** ≤ 10000).

```
Algorithm ceFace(n):
      s ← 0
      While n > 0 execute
            c ← n MOD 10
            If  (c MOD 3 = 1) OR (c MOD 3 = 2)
                  then
                        s ← s + (c MOD 3)
            EndIf
            n ← n DIV 10
      EndWhile
      return s
EndAlgorithm
```

Care afirmaţii sunt **adevărate**?

A. Pentru n=987654 se returnează 5;
B. Pentru n=987654 se returnează 6;
C. Pentru n=7777777 se returnează 7;
D. Pentru n=8888888 se returnează 8.

**1.125.** Se consideră toate șirurile de lungime ***L*** ϵ {1, 2, 3} formate din litere din mulțimea alfabetului englez (26 litere din care 5 vocale)
Câte dintre aceste șiruri au elementele ordonate strict crescător (conform alfabetului) și un număr impar de vocale? (*a, e, i, o, u* sunt vocale)

**A.** 1160
**B.** 1165
**C.** 1170
**D.** 1175

**1.126.** Precizați care dintre următoarele expresii are valoarea adevărat dacă și numai dacă numărul natural ***n*** este divizibil cu 6 și are ultima cifră 2 sau 8:

**A.** `(n MOD 9 = 0) AND ((n MOD 10 = 4) OR (n MOD 10 = 8))`
**B.** `(n MOD 6 = 0) AND ((n MOD 10 = 2) OR (n MOD 10 = 8))`
**C.** `((n MOD 3 = 0) AND (n MOD 10 = 2)) OR ((n MOD 3 = 0) AND (n MOD 10 = 8))`
**D.** `(n MOD 3 = 0) AND (n MOD 10 = 2) AND (n MOD 10 = 8)`

**1.127.** Se consideră toate șirurile de lungime ***L*** ϵ {1, 2, 3} formate din litere din mulțimea alfabetului englez (26 litere din care 5 vocale)
Câte dintre aceste șiruri sunt palindroame și au un număr impar de vocale? (*a, e, i, o, u* sunt vocale).

**A.** 115
**B.** 125
**C.** 135
**D.** 145

**1.128.** Se consideră algoritmul **calcul(x, n)**, unde parametrii de intrare sunt numerele naturale ***n*** și ***x***, având proprietatea: **1 ≤ *x* ≤ *n* < 10**.

```
Algorithm calcul(x, n):
        b ← 1
        For i ← 1, n – x, 2  execute
                b ← b + i
        EndFor
        a ← b – 1                                   (*)
        For i ← n – x + 1, n, 2  execute
                a ← a + i
        EndFor
        return a - b
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Dacă ***n*** = **7** și ***x*** = **3**, atunci algoritmul returnează **12**.
B. Dacă ***n*** = **7** și ***x*** = **3**, atunci algoritmul returnează **11**.
C. Pentru orice valori ale lui **x** şi **n** (cu precondiţii adevărate) **a** se iniţializează cu **0**. (*)
D. Algoritmul returnează întotdeauna o valoare strict mai mare decât **0**.

**1.129.** Se dă algoritmul s(a, b, c), unde ***a***, ***b***, ***c*** sunt numere naturale pozitive (1 ≤ ***a***, ***b***, ***c*** ≤ 10000).

```
Algorithm s(a, b, c):
        If (a = 0 ) OR (b = 0) OR (c = 0) then
                return 1
        else
                If a > b then
                        return  a * s(a - 1, b, c)
                else
                        If a < b then
                                return  b * s(a, b - 1, c)
                        else
                                return  c * s(a - 1, b - 1, c - 1)
                        EndIf
                EndIf
        EndIf
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate în cazul în care ***a*** = ***b*** și ***a*** < ***c***;

A. Algoritmul calculează și returnează ***c***! / (***c*** - ***a***)!
B. Algoritmul calculează și returnează ***c***! / (***c*** – ***a*** + 1)!
C. Algoritmul calculează și returnează ***c***! / (***c*** – ***a*** - 1)!
D. Algoritmul calculează și returnează aranjamente de ***c*** luate câte ***a*** .

**1.130.** Se dă algoritmul s(a, b, c), unde ***a***, ***b***, ***c*** sunt numere naturale pozitive (1 ≤ ***a***, ***b***, ***c*** ≤ 10000).

```
Algorithm s(a, b, c):
    If (a = 0 ) OR (b = 0) OR (c = 0) then
        return 1
    else
        If a > b then
            return  a * s(a - 1, b, c)
        else
            If a < b then
                return  b * s(a, b - 1, c)
            else
                return  c * s(a - 1, b - 1, c - 1)
            EndIf
        EndIf
    EndIf
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt adevărate în cazul în care ***a*** < ***b*** < ***c***;

A. Algoritmul calculează și returnează (b!/(b-a)!) * (***c***! / (***c*** - ***a***)!);
B. Algoritmul calculează și returnează (b!/(b-a+1)!) * (***c***! / (***c*** - ***a***)!);
C. Algoritmul calculează și returnează (b!/(b-a-1)!) * (***c***! / (***c*** - ***a***)!);
D. Algoritmul calculează și returnează valoarea 4200 pentru a=3, b=5, c=7.

**1.131.** Fie ***s*** un șir de numere naturale unde elementele $s_i$ sunt de forma

$$s_i = \begin{cases} x, \text{ dacă } i = 1 \\ x + 1, \text{ dacă } i = 2 \\ s_{i-2} \ @ \ s_{i-1} \text{ dacă } i > 2 \end{cases} \qquad (i = 1, 2, ...).$$

Operatorul @ concatenează cifrele operandului stâng cu cifrele operandului drept, în această ordine (cifre aferente reprezentării în baza 10), iar ***x*** este un număr natural (1 ≤ ***x*** ≤ 99). De exemplu, dacă ***x*** = 2, șirul ***s*** va conține valorile 2, 3, 23, 323, 23323, ...
Pentru un numar natural ***k*** (1 ≤ ***k*** ≤ 30) precizați numărul cifrelor acelui termen din șirul ***s*** care este succesorul termenului format din ***k1*** cifre, unde ***k1*** este cel mai mare număr cu proprietatea că ***k*** ≤ ***k1*** ≤ 30 și există un termen format din ***k1*** cifre.

A. dacă ***x*** = 15 și ***k*** = 8, numărul cifrelor termenului căutat este 42.
B. dacă ***x*** = 2 și ***k*** = 6, numărul cifrelor termenului căutat este 55.
C. dacă ***x*** = 14 și ***k*** = 27, numărul cifrelor termenului căutat este 26.
D. dacă ***x*** = 5 și ***k*** = 12, numărul cifrelor termenului căutat este 34.

**1.132. Ce returnează algoritmul m2(n)? Parametrul n este natural. (sursă: personală)**

```
Algoritmul m2(n):
   rez← 0
   nr ← 0
   backN← n
   Pentru c← 0,9 execută
      n← backN
      CâtTimp (n>0)
         Dacă(n Mod 10 = c) AND (c=0)
            Atunci
                nr← nr+1
            Altfel
                Dacă (n Mod 10 = c and c>0)
                   Atunci
                      rez← rez*10+c;
                      CâtTimp(nr>0)
                            rez← rez*10;
                            nr ← nr-1
                      SfCâtTimp
               SfDacă
         SfDacă
         n←n/10
      SfCâtTimp
  SfPentru
  Return rez
SfAlgoritm
```

**A.** dacă n=303, returnează 303
**B.** dacă n=30303, returnează 30303
**C.** dacă n=60606, returnează 60606
**D.** dacă n=90909, returnează 90099

# 2. Vectori de 1 dimensiune (şiruri)

**2.1.** Ce returnează algoritmul **s(n,v), n>1**, natural şi **v[1],...,v[n]** un vector cu numere întregi? (sursa:personală)

```
Algorithm prim(n)
...
      {returnează 1, dacă n e prim}
      {returnează 0, dacă n nu e prim}
EndAlgorithm

Algorithm s(n,v):
  s1← 0
  s2← 0
  For i← 1,n execute
      If (prim(i)=0) AND  (prim(v[i])=1)
         then
              s1← s1+v[i]
         else
              s2← s2+v[i]
     EndIf
 EndFor
 return s1-s2
EndAlgorithm
```

**A.** Diferenţa dintre suma valorilor prime din vector şi suma valorilor neprime;
**B.** Diferenţa dintre suma numerelor prime de pe poziţiile neprime din vector şi suma celorlate valori din vector;
**C.** Diferenţa dintre suma numerelor prime de pe poziţiile neprime din vector şi suma valorilor neprime de pe poziţiile prime din vector;
**D.** Pentru **n=6, v = [ 1, 2, 14, 13, 15, 7 ]** se returnează **-12.**

**2.2.** Se consideră subalgoritmul **dif(a, n),** unde **a** este un șir cu **n** numere întregi **(n –** număr natural**, 1 < n < 100):** (sursa: admitere/concurs FMI UBB Cluj-Napoca)

```
Algorithm dif(a, n):
     If n = 0
       then
           return 0
     EndIf
     If |a[n]| MOD 2 = 0              { |a[n]| e modulul lui a[n] }
       then
           return dif(a, n - 1) + a[n]
       else
           return dif(a, n - 1) - a[n]
     EndIf
EndAlgorithm
```

Precizați pentru care valori ale lui **n** și **a** subalgoritmul returnează valoarea **0.**

**A. n = 4 și a = ( 6, 4, 5, 5)**
**B. n = 4 și a = (-6, 5, 4, -7)**
**C. n = 8 și a = (-6, 5, -1, -4, 1, 4, -7, 6)**
**D. n = 8 și a = (-6, -3, 0, 1, 2, 3, -1, 4)**

**2.3.** Ce face algoritmul ? Vectorul **x** are valori întregi şi are lungimea **n** (date de intrare). Vectorul **p** şi lungimea sa **k** trebuie determinate (sursa:personală)

```
Algorithm pm(x, n){
  k   ← 1
  p[1]← 1
  For i← 2,n execute
     If x[i] > x[p[1]]
          then
               k   ← 1
               p[1]← i
          else
                If x[i] = x[p[1]]
                   then
                           k   ← k+1
                           p[k]← i
                EndIf
     EndIf
  EndFor
EndAlgorithm
```

**A.** Pentru **x=(9, 8, 7, 6, 5, 4, 3, 2, 1) ⇒ P=(1,2,3);**
**B.** Pentru **x=(1, 3, 9, 4, 6, 5, 8, 7, 9) ⇒ P=(1,3,9);**
**C.** Pentru **x=(4, 8, 3, 5, 8, 2, 8, 5, 1) ⇒ P=(2,5,7);**
**D.** Pentru **x=(8, 4, 3, 2, 8, 2, 5, 8, 3) ⇒ P=(4,6).**

**2.4.** Ce returnează algoritmul **a(v,n)**? **v** este un vector de numere întregi, iar **n** este lungimea lui **v**. (sursa:personală)

```
Algorithm a(v,n):          { algoritmul prim(m) returneaza 1 daca m e prim şi }
                           { 0 în caz contrar }
  k← 1
  While (k ≤ n) AND (prim(v[k])=0) execute
      k← k + 1
  EndWhile
  If k > n
     then
           returnează -1
     else
           m← v[k]
           k← k + 1
           While k ≤ n execute
               If m = v[k]
                 then
                       returnează 2
                 else
                       k← k + 1
               EndIf
           SfCătTimp
  EndIf
  return 1
EndAlgorithm
```

**A.** For **v= (9, 8, 7, 6, 5, 4, 3, 2, 9)** ⇒ **-1**
**B.** For **v= (1, 3, 9, 4, 6, 5, 8, 7, 9)** ⇒ **2;**
**C.** For **v= (4, 8, 3, 5, 8, 2, 8, 5, 1)** ⇒ **1;**
**D.** For **v= (8, 4, 3, 2, 8, 2, 5, 8, 3)** ⇒ **2.**

**2.5.** Avem 4 variante ale algoritmului de căutare binară. Care e corectă? (**x** este, aprioric, sortat crescător). (sursa:personală)

```
A.{ls=1, ld=lung.lui x la apelul primar}
Algorithm cbr(ls, ld, x, val):
  m← (ls+ld) DIV 2
  If ls ≤ ld
    then
      If val = x[m]
        then
            return m
      EndIf
      If val < x[m]
        then
            return cbr(ls,m-1,x,val)
        else
            return cbr(m+1,ld,x,val)
      EndIf
  EndIf
  return -1
EndAlgorithm
```

```
B. {ls=1, ld=lung.lui x la apelul primar}
Algorithm cbr(ls, ld, x, val):
  m← (ls+ld) DIV 2
  If ls ≤ ld
    then
      If val = x[m]
        then
            return m
      EndIf
      If val < x[m]
        then
            return cbr(ls,m+1,x,val)
        else
            return cbr(m-1,ld,x,val)
      EndIf
  EndIf
  return -1
EndAlgorithm
```

```
C. {n lungimea lui x}
Algorithm cbr(n, x, val):
  ls← 1
  ld← n
  CâTimp ls ≤ ld  execute
     m← (ls+ld) DIV 2
     If val=x[m]
        then
                  return m
        EndIf
        If val < x[m]
          then
                  ld ←m
          else
                  ls ← m-1
        EndIf
  EndWhile
  return -1
EndAlgorithm
```

```
D. {n lungimea lui x}
Algorithm cbr(n, x, val):
  ls← 1
  ld← n
  CâTimp ls ≤ ld  execute
     m← (ls+ld) DIV 2
     If val=x[m]
        then
            return m
     EndIf
     If val < x[m]
        then
             ld ← m-1
        else
             ls ← m+1
     EndIf
  EndWhile
  return -1;
EndAlgorithm
```

**2.6.** Ce valori au variabilele **ls**, **ld**, fie la terminarea instrucţiunii repetitive **Repetă ... PânăCand** fie înainte de instrucţiunea return m? (sursa: personală)

```
Algorithm cautBinar(n, x, val): {n lungimea lui x}
    ls← 1
    ld← n
    Repetă
      m← (ls+ld) DIV 2
      If x[m] = val
        then
                (*)return m
      EndIf
      If val < x[m]
        then
                ld← m-1
        else
                ls← m+1
      EndIf
    Until ls > ld
    return -1
EndAlgorithm
```

**A.** n=6, x={1,2,3,4,5,6}, val = 10 => ls=7, ld=8
**B.** n=6, x={1,2,3,4,5,6}, val = 10 => ls=7, ld=5
**C.** n=6, x={1,2,3,4,5,6}, val = 4 => ls=5, ld=6
**D.** n=6, x={1,2,3,4,5,6}, val = 4 => ls=4, ld=4

**2.7.** Vectorul **p**, de **n** componente (**n>10**), este modificat după secvenţa de mai jos. Aprioric v este iniţializat cu **0**. (sursa: personală). Care afirmaţii sunt adevarate?

```
For i← 2,[√n] execute
  For k← 2*i,n,i execute
        p[k]← 1
  EndFor
EndFor
```

**A.** Pentru fiecare **j** par şi **(1 ≤ j ≤ n)** atunci **P [ j ] = 1**;
**B.** Pentru fiecare **j impar** şi **(1 ≤ j ≤ n)** atunci **P [ j ] = 0**;
**C.** Dacă **j** este prim şi **(1 ≤ j ≤ n)** atunci **P [ j ] = 1**;
**D.** Dacă **j** este prim şi **(1 ≤ j ≤ n)** atunci **P [ j ] = 0**.

**2.8.** Ce face algoritmul **x(v,n)**? **v** este un vector de n numere întregi. (sursa:personală)

```
Algorithm x(v,n): {n lungimea lui v}
  For k← 1,n-1 execute
    For i← 1,n-1 execute
      If v[i] < v[i+1]
        then
          v[i]←→ v[i+1] { interschimba }
      EndIf
    EndFor
  EndFor
EndAlgorithm
```

A. BubbleSort crescator vectorul **v**;
B. SelectSort descrescător vectorul **v**;
C. SelectSort crescător vectorul **v**;
D. BubbleSort descrescator vectorul **v**;

**2.9.** Ce face algoritmul **x(v,n)**? **v** este un vector de **n** numere întregi (**n>3**). (sursa:personală)

```
Algorithm x(v,n): {n lungimea lui v}
  For k← 1,3 execute
    For i← 1,n-1 execute
      If v[i] > v[i+1]
        then
          v[i]←→ v[i+1] { interschimba }
      EndIf
    EndFor
  EndFor
EndAlgorithm
```

**A.** BubbleSort crescător a vectorului **v**;
**B**. Pe primele **3** poziţii se vor afla cele mai mici **3** valori ale vectorului **v** şi acestea sunt sortate crescător;
**C**. Pe ultimele **3** poziţii se vor afla cele mai mari **3** valori ale vectorului **v** şi acestea sunt sortate crescător;
**D**. Sortează crescător vectorul **v**, pentru **n=4**.

**2.10.** Ce fac cei doi algoritmi? (v este un vector de **n** componente întregi, **n>3**). (sursa:personală)

```
Algorithm b1(n,v):
  For k← 1,n-1 execute
      For i← 1,n-1 execute
           If v[n-i+1] < v[n-i]
              then
                 v[n-i+1] ←→ v[n-i] { interschimba }
           EndIf
      EndFor
  EndFor
EndAlgorithm

Algorithm b2(n,v):
  For k← 1,n-1 execute
      For i← n,2,-1 execute
           If v[i] < v[i-1]
              then
                 v[i] ←→ v[i-1] { interschimba }
           EndIf
      EndFor
  EndFor
EndAlgorithm
```

**A.** **b1** sortează crescător pe **v, b2** sortează descrescator pe **v;**
**B.** **b1** sortează descrescător pe **v, b2** sortează crescator pe **v;**
**C.** **b1** sortează descrescător pe **v, b2** sortează descrescator pe **v;**
**D.** **b1** sortează crescător pe **v, b2** sortează crescator pe **v.**

**2.11.** Fie algoritmul de mai jos, care variante sunt adevărate? (sursa:personală)

```
Algorithm x(v,n): {n lungimea lui v}
  For k← 1,n-1 execute
    For i← 1,n-1 execute
      If v[i] > v[i+1]
        then
          v[i]←→ v[i+1] { interschimba }
      EndIf
    EndFor
  EndFor
EndAlgorithm
```

**A**.Înlocuind If ... EndIf, cu instrucţiunile:

```
d← i+1
If v[i] > v[i+d]
   then
     v[i]←→ v[i+d]
EndIf
```

Efectul algoritmului se schimbă.

**B**. Înlocuind If ... EndIf, cu instrucţiunile:

```
d← i+1
If v[i] > v[d]
   then
     v[i]←→ v[d]
EndIf
```

**Efectul algoritmului nu se schimbă.**

**C**. Înlocuind If ... EndIf, cu:

```
If v[n-i+1] < v[n-i]
   then
     v[n-i]←→v[n-i+1]
EndIf
```

**Efectul algoritmului nu se schimbă.**

**D**. Înlocuind If ... EndIf, cu:

```
If v[n-i+1] > v[n-i]
   then
     v[n-i]←→v[n-i+1]
EndIf
```

**Efectul algoritmului nu se schimbă.**

**2.12.** Se dă o variantă a tehnicii bulelor de sortare a vectorului **v** de lungime **n (n>1)**. (sursa:personală)

```
Algorithm b(n v):
   poz← n       {poz = poziţia ultimei schimbări după instrucţiunea For}
   Repetă
     ok← 1                        {ok e variabilă semafor}
     pI← 1                        {pI e poziţia unei schimbări}
     For i← 1,poz-1 execute
        If  v[i] > v[i+1]
          then
               v[i] ←→ v[i+1]    (*)
               ok← 0
               pI← i
          EndIf
     EndFor
     poz← pI
   PănăCând ok = 1
EndAlgorithm
```

De câte ori se execute interschimbarea (*), pentru **n=8** şi **v= {3, -9, 123, 17, -333, -56, 7, 27}**?

**A.** de 12 ori;

**B.** de 13 ori;

**C.** de 14 ori;

**D.** de 15 ori.

**2.13.** Se dă o variantă a tehnicii selecţiei de sortare a vectorului v de lungime n (n>1). (sursa:personală)

```
Algorithm s2Cresc (n, v):
   For i← 1,n-1
        pMin← i
        For j←i+1,n
            If v[j] < v[pMin]
               then
                  pMin← j
            EndIf
        EndFor
        v[i] ←→ v[pMin]  {interschimbare}
   EndFor
EndAlgorithm
```

**De câte ori se face interschibarea pentru n=8 şi v={3,-9,123,17,-333,-56,7,27}**

**A.** de 9 ori
**B.** de 8 ori
**C.** de 7 ori
**D.** de 6 ori

**2.14.** Se dă o variantă a tehnicii inserţiei de sortare a vectorului **v** de lungime **n.** (sursa:personală)

```
Algorithm insCresc(n,v):
   For i← 2,n execute
      aux← v[i]
      j  ← i-1
      While (j > 0) AND (aux < v[j]) execute
            v[j+1] ← v[j]                    (*)
            j      ← j-1
      EndWhile
      v[j+1] ← aux;
   EndFor
EndAlgorithm
```

De câte ori se execute instrucţiunea **(*)** pentru **n=8** şi **v={3,-9,123,17,-333,-56,7,27}?**

**A.** de 12 ori
**B.** de 13 ori
**C.** de 14 ori
**D.** de 15 ori

**2.15.** Care afirmaţii sunt adevărate, pentru cei 4 algoritmi? (sursa:personală)

**A**. Algoritmul **1** şi algoritmul **2** fac sortarea prin algoritmul bulelor;
**B**. Algoritmul **3** şi algoritmul **4** fac sortarea prin inserţie;
**C**. Algoritmul **4** face sortarea prin selecţie;
**D**. Algoritmul **3** face sortarea prin inserţie;

```
1.    {v vector iar n=lungimea  v}
   Algorithm unu(n,v):
   For k← 1,n-1  execute
     For i k← 1,n-1 execute
       If v[i] > v[i+1]
        then              {schimbare}
           v[i]←→ v[i+1]
       EndIf
     EndFor
   EndFor
 EndAlgorithm
```

```
2.    {v vector iar n=lungimea  v}
      {apelul primar cu i=1, ok=true}
Algorithm doi(n,x,i,ok):
  If i < n
    then
    If x[i] > x[i+1]
      then
            v[i]←→ v[i+1] {schimbare}
            ok← false;
    EndIf
    doi(n,x,i+1,ok)
  EndIf
  If ok = false
      then
            doi(n,x,i←1,ok←true)
  EndIf
EndAlgorithm
```

```
3.   {x vector iar n=lungimea lui x}
  {apelul primar cu i←2,j←1,aux←x[2]}
Algorithm trei(n,i,j,x,aux):
  If i ≤ n
    then
       If (j > 0) AND (aux < x[j])
          then
            x[j+1]← x[j]
            trei(n,i,j-1,x,aux)

          else
            x[j+1]← aux
            trei(n,i+1,i,x,x[i+1])
       EndIf
  EndIf
EndAlgorithm
```

```
4.    {v vector iar n=lungimea  v }
      {apelul primar D(n,v,i←1,j←2)}
Algorithm patru(n,v,i,j):
  If i < n
    then
      If  j ≤ n
        then
          If v[i] > v[j]
            then
               v[i]←→ v[j] {schimbare}
          EndIf
          patru(n,v,i,j+1)
        else
          patru(n,v,i+1,i+2)
      EndIf
  EndIf
EndAlgorithm
```

**2.16.** Fie algoritmul **f** de mai jos. Cum se aranjează elementele vectorului **a** (indicii de la **1** la **n**), după execuţia algoritmului (sursa:personală)

```
Algorithm f(n,a):
  i← j← 1
  While j ≤ n execute
    If a[j] ≥ 0
      then
          j← j+1
      else
          a[j]←→ a[i]
          i   ← i+1
          j   ← j+1
    EndIf
  EndWhile
EndAlgorithm
```

**A.** Primele elemente sunt **> 0** apoi ultimele sunt **≤ 0**;
**B.** Primele elemente sunt **≤ 0**;
**C.** Există **k∈{1,...,n}** astfel că primele **k** sunt amestecate, iar ultimele **(n-k) ≥ 0**;
**D.** Există **k∈{0,...,n}** astfel că primele **k** sunt **< 0**, iar ultimele **(n-k) ≥ 0**.

**2.17.** Ce valori au variabilele iM şi Im, după executarea secvenţei? (sursa:personală)

```
 {x e vector, n e lungimea lui x}
Im← iM← n
For i← n-1,1,-1 execute
  If x[i] > x[Im]
    then
          Im← i
  EndIf
  If x[i] < x[iM])
    then
          iM← i
  EndIf
EndFor
```

**A**. Pentru **x=(9,8,7,6,5,4,3,2,1) ⇒ Im=9, iM=1;**
**B**. Pentru **x=(9,8,7,6,5,4,3,2,1) ⇒ Im=5, iM=5;**
**C**. Pentru **x=(9,8,7,6,5,4,3,2,1) ⇒ Im=1, iM=9;**
**D**. Pentru **x=(9,8,7,6,5,4,3,2,1) ⇒ Im=9, iM=9;**

**2.18.** Ce face Algoritmul următor **pmn (n,x)**? (**n** natural şi este lungimea vectorului **x**). Tabloul **p** şi lungimea lui **k** se crează. (sursa:personală)

```
Algorithm pmn(n,x):  {n şi x date, n>1}
   i← 1
  While (i ≤ n) AND (x[i] ≥ 0) execute
            i← i+1
  EndWhile
  If i ≤ n
    then
       k←1
       p[k]← i
       For j← i+1,n execute
          If x[j] = x[p[1]]
            then
                 k← k+1
                 p[k] ←j
            else
                 If (x[j] < 0) AND (x[j] > x[p[1]])
                    then
                       k← 1
                       p[k] ←j
                 EndIf
           EndIf
        EndFor
    else
       k← 0
  EndIf
EndAlgorithm
```

**A.** Determină poziţiile numerelor negative din **x** sau **k=0** în caz contrar;
**B.** Determină valorile numerelor negative din **x** sau **k=0** în caz contrar;
**C.** Determină poziţiile numărului maxim negativ din **x**, sau **k=0** în caz contrar;
**D.** Determină valoarea numărului maxim negativ din **x**, sau **k=0** în caz contrar.

**2.19.** Ce conţine vectorul **y** (**n**>1, natural şi este lungimea vectorilor **x** şi **y**). Algoritmul **creareY**(...) apelează algoritmul **mm(...)**.

```
Algorithm creareY(n,x,y):
    For i← 1,n execute
          y[i]← mm(n,x,i)
    EndFor
EndAlgorithm

Algorithmm mm (n,x,i):
    cnt← 0
    j  ← 1
    CătTimp  j < i  execute
       If x[j] < x[i]
         then
               cnt← cnt+1
       EndIf
       j← j+1
    EndWhile
    return cnt
 EndAlgorithm
```

**A. x= {1, 2, 3, 4, 5, 6} => y= {0, 1, 2, 3, 4, 5};**
**B. x= {1, 2, 3, 4, 5, 6} => y= {5, 4, 3, 2, 1, 0};**
**C. x= {1, 20, 3, 40, 5, 6} => y= {0, 1, 1, 3, 2, 3};**
**D. x= {1, 2, 3, 4, 5, 6} => y= {1, 1, 1, 1, 1, 1}.**

**2.20.** Algoritmul **creY(...)** apelează, ceilalţi doi algoritmi; **m,n>1** naturale; vectorul **x** şi lungimea lui **n**, sunt date; trebuie creat vectorul **y** şi lungimea lui **m**.

```
Algorithm creY(n, x, m, y):
    m← 0
    For i← 1,n execute
       poz← existaInY(m,y,x[i])
       If poz = 0
         then
            adaugInY(m,y,x[i])
       EndIf
    EndFor
 EndAlgorithm

Algorithm existaInY(m, y, v)
   For i← 1,m execute
      If v = y[i]
         then
            returnează i
       EndIf
   EndFor
   returnează 0
 EndAlgorithm

Algorithm adaugInY (m, y, v)     {m transmis prin referinţă}
   j← m
   CatTimp (j > 0) AND  (v < y[j]) execute
       y[j+1]← y[j]
       j     ← j-1
    EndWhile
    y[j+1] ← v
    m← m+1
 EndAlgorithm
```

**A.** Dacă **x= {1, 2, 3, 4, 5, 1, 2, 3, 4}, n=9 => y= {5, 4, 3, 2, 1} şi m=5;**
**B.** Dacă **x= {1, 2, 30, 2, 5, 1, 2, 5, 4}, n=9 => y= {1, 2, 30, 5, 4} şi m=5;**
**C.** Dacă **x= {1, 2, 30, 2, 5, 1, 2, 5, 4}, n=9 => y= {1, 2, 4, 5, 30} şi m=5;**
**D.** Dacă **x= {1, 1, 1, 1, 1, 1}, n=6 => Y= {1} şi m=1;**

**2.21.** Algoritmul **cmls(...)** apelează algoritmul **s(...).** Se dau: vectorul **x** şi **n>0** lungimea lui **x**; **st** şi **dr** trebuie determinate. Care afirmaţii sunt corecte?

```
Algorithm cmls(n, x):
      i ← 1
      st← 0
      dr← -1
      While i < n execute
            While (i < n) AND (x[i] ≥ x[i+1]) execute
                  i← i+1
            EndWhile
            If (i < n)
               then
                 j← s(i,n,x)
                 If (j-i > 0) AND (j-i > dr-st)
                    then
                       st← i
                       dr← j
                 EndIf
            EndIf
            i← j+1
      EndWhile
EndAlgorithm

Algorithm s(i, n, x):
      j← i
      While (j < n) AND (X[j] < X[j+1]) execute
            j← j+1
      SfCătTimp
      returnează j
EndAlgorithm
```

**A.** Dacă **X= {1, 2, 3, -5, 3, 1, 2, 3, 4}, n=9** ⇒ **st = 1, dr = 3**
**B.** Dacă **X= {1, 2, 30, 2, 5, 1, 2, 5, 10}, n=9** ⇒ **st = 6, dr = 9;**
**C.** Dacă **X= {1, 2, 30, 2, 5, 1, 2, 5, 4}, n=9** ⇒ **st = 6, dr = 8;**
**D.** Dacă **X= {1, 1, 1, 1, 1, 1}, n=6** ⇒ **st = 0, dr = -1;**

**2.22.** Algoritmul **dp(...)** are ca parametri vectorul **x** (valori nenule) şi **n>0** lungimea lui **x**;
Vectorul **p** trebuie determinat, **x** are valori întregi nenule. Care afirmaţii sunt adevărate?

```
Algorithm dp (n, x){
      For i← 1,n
            p[i] ← i
      EndFor
      For i← 1,n-1 execute
          For j← i+1,n execute
               If x[p[i]] < x[p[j]]
                   then
                        p[i] ← p[i] * p[j]
                        p[j] ← p[i] DIV p[j]
                        p[i] ← p[i] DIV p[j]
                EndIf
           EndFor
       EndFor
EndAlgorithm
```

**A.** Dacă **X= {1, 2, 3, -5, 6, -7}, n=6 => P= {1, 2, 3, 4, 5, 6};**
**B.** Dacă **X= {1, 2, 30, 2, 8, 1}, n=6 => P= {3, 5, 4, 2, 1, 6};**
**C.** Dacă **X= {1, 2, 30, 2, 5, 1}, n=6 => P= {1, 6, 2, 4, 5, 3};**
**D.** Dacă **X= {1, 1, 1, 1, 1, 1}, n=6 => P= {1, 2, 3, 4, 5, 6}.**

**2.23.** Algoritmul **fda**(..) schimbă ordinea elementelor iniţiale din vectorul **x** de lungime **n**, **a** este dat de asemenea.
Care afirmaţii sunt adevărate?

```
Algorithm fda(n,x,a):
    i← 1
    j← 1
    While  j ≤ n execute
       If x[j] < a
            then
                x[j]←→x[i]
                i← i+1
       EndIf
       j← j+1
    EndWhile
EndAlgorithm
```

**A.** Dacă **X= { 1, 2, 3, -5, 6, -7 }, n=6, a= 4 => X= { 1, 2, 3, -5, -7, 6 }**
**B.** Dacă **X= { 1, 2, 30, 2, 5, 1 }, n=6, a= 4 => X= { 1, 2, 2, 1, 5, 30 };**
**C.** Dacă **X= { 1, 2, 30, 2, 5, 1 }, n=6, a= - 4 => X= { 2, 1, 30, 5, 2, 1 };**
**D.** Dacă **X= { 1, 2, 5, 9, 10, 20 },n=6, a= 5 => X= { 20, 1, 2, 5, 9, 10 };**

**2.24.** Algoritmul **ea(...)** modifică vectorul **x** de lungime **n**, **a** este dat de asemenea.
Care afirmaţii sunt adevărate?

```
Algorithm ea (n,x,a):
     i← 1
     While  i ≤  n  execute
          If x[i] ≤ a
               then
                  For j← i,n-1 execute
                      x[j]← x[j+1]
                  EndFor
                  n← n-1
               else
                  i← i+1
          EndIf
     EndWhile
EndAlgorithm
```

**A.** Dacă **X= {1, 2, 3, -5, 6, -7}, n=6, a=4 => X= {1, 2, 3, -5, -7}, n=5**
**B.** Dacă **X= {1, 2, 30, 2, 5, 1}, n=6, a=2 => X= {30, 5}, n=2;**
**C.** Dacă **X= {1, 2, 30, 2, 5, 1}, n=6, a=-4 => X= {2, 1, 30, 5, 2, 1}, n=6;**
**D.** Dacă **X= {1, 2, 5, 9, 10, 20}, n=6, a=5 => X= {20, 1, 2, 5, 9, 10}, n=6;**

**2.25.** Se dau n, natural n>1, şi vectorul de întregi x. Ce valori are x după execuţia algoritmului ce urmează (alegeţi variantele corecte):

```
Algorithm selectie (n, x):
   For i← 1,3 execute
     For j←  i+1,n execute
       If x[i] > x[j]
         then
            x[i] ←→ x[j]
       EndIf
     EndFor
   EndFor
SfAlgorithm
```

**A. n=6, x= {4, 5, 6, 3, 2, 1} ⇒ x = {1, 2, 3, 4, 5, 6}**
**B. n=2, x= {5, 4} ⇒ x = {4, 5}**
**C. n=5, x= {5, 4, 3, 1, 2} ⇒ x = {1, 2, 3, 4, 5}**
**D. n=5, x= {5, 4, 3, 1, 2} ⇒ x = {1, 2, 3, 5, 4}**

**2.26.** Se dau algoritmii **pc(...)** şi **ps(...)** . Algoritmul **ps(...)** are parametrii de intrare un vector **v** şi doi indici **st**, **dr**. Apelul lui **ps** (din exterior) se face cu **st=1** şi **dr=**lungimea lui **v**. Algoritmul **ps** apelează algortimul **pc(...).**

```
Algorithm ps(v, st, dr):
       i← st
       d← st+1
       While  d ≤ dr execute
          If  v[i] ≤ v[d]
            then
                 d← d+1
            else
                pc(v,i,d)
                i← i+1
                d← d+1
          EndIf
       EndWhile
       return i
EndAlgorithm

Algorithm pc(v, p, q)                          {p<q}
      aux← v[q]
      For i← q,p+1,-1 execute
            v[i] ← v[i-1]
      EndFor
      v[p] ← aux
EndAlgorithm
```

Care afirmaţii sunt adevărate?

- **A.** Dacă **v= {-10, 10, -10, 10, -10, 10}, n=6,** algoritmul **pc** returnează **4;**
- **B.** Dacă **v= {-14, 13, -12, 11, -10, 9}, n=6,** algoritmul **pc** returnează **2;**
- **C.** Dacă **v= {-9, 10, -11, 12, -13, 14}, n=6,** algoritmul **pc** returnează **3;**
- **D.** Dacă **v= {9, 3, 15, 12, -100, 2), n=6,** algoritmul **pc** returnează **4.**

**2.27.** Se dă algoritmul **pU(...)** de mai jos. Algoritmul **pu** are parametrii de intrare un vector **x** şi doi indici **st**, **dr**. Apelul lui **pU** (din exterior) se face cu **st=1** şi **dr=lungimea** lui **v**.

```
Algorithm pU(x, st, dr):
  i← st
  For j← st,dr execute
    If x[j] < x[dr]
       then
            x[i] ←→x[j]
            i     ← i+1
    EndIf
  EndFor
  x[i] ←→x[dr]
  return i
EndAlgorithm
```

Care afirmaţii sunt adevărate?

**A.** Pentru **V= {-10, 10, -10, 10, -10, 10}, n=6,** algoritmul **pU** returnează **4;**
**B.** Pentru **V= {-14, 13, -12, 11, -10, 9}, n=6,** algoritmul **pU** returnează **1;**
**C.** Pentru **V= {-9, 10, -11, 12, -13, 14}, n=6,** algoritmul **pU** returnează **3;**
**D.** Pentru **V= {9, 3, 15, 12, -100, 2), n=6,** algoritmul **pU** returnează **2.**

**2.28.** Se dă algoritmul **ps(...)** de mai jos. Algoritmul **ps** are parametrii de intrare un vector **x** şi doi indici **st**, **dr**. Apelul lui **ps** (din exterior) se face cu **st=1** şi **dr=lungimea** lui x.

```
Algorithm ps(x, st, dr):
       i← dr
       For j← dr,st,-1  execute
            If  x[j] > x[st]
               then
                  x[i] ←→x[j]
                  i    ← i-1
            EndIf
       EndFor
       x[i] ←→x[st]
       return i
EndAlgorithm
```

Care afirmaţii sunt adevărate?

**A.** Pentru **V = { -10, 10, -11, 10, -12, 10}, n=6,** algoritmul **ps** returnează **4;**
**B.** Pentru **V = {-1, 13, -12, 11, -10, 9}, n=6,** algoritmul **ps** returnează **3;**
**C.** Pentru **V = {-9, 10, -11, 12, -13, 14}, n=6,** algoritmul **ps** returnează **3;**
**D.** Pentru **V = {-9, 3, 15, 12, -100, 2), n=6,** algoritmul **ps** returnează **4.**

**2.29.** Ce returnează algoritmul **func**. Vectorul **f** are valorile unei funcţii definite pe **{1..m}** cu valori in **{1..n}**. Exemplu: o funcţie: **{1, 2, 3}** $\rightarrow$ **{1, 2, 3, 4}** este reprezentată de vectorul **f = {2, 4, 3}.**

**E funcţie**

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 4 | 3 |

**f injectiva**

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 3 | 4 |

**Nu e funcţie**

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 7 | 3 |

**f: {1,2,3,4} $\rightarrow$ {1,2} se pot construi f. surjective**

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 2 | 1 | 2 | 2 |

```
Algorithm func (m, f, n):
   If esteFunctie(m,f,n) = false
     then
       return false
     else
       return true
   EndIf
EndAlgorithm

Algorithm esteFunctie(m, f, n):
    For i← 1,m   execute
      If (f[i] < 1) OR (f[i] > n)
        then
           return false
      EndIf
    EndFor
    return true
EndAlgorithm
```

**A. false** dacă **f** este funcţie bijectivă;
**B. false** dacă **f** este funcţie surjectivă;
**C. true** dacă **f** este functie;
**D. false** dacă **f** este funcţie injectivă.

**2.30.** Ce returnează algoritmul **func**. Vectorul **f** are valorile unei funcţii definite pe **{1..m}** cu valori în **{1..n}**.
Exemplu: o funcţie: **{1, 2, 3}** $\rightarrow$ **{1, 2, 3, 4}** este reprezentată de vectorul **f = {2, 4, 3}.**

```
Algorithm func (m, f, n):
   If m ≠ n
     then
        returnează false
   EndIf
   If esteFunctie(m,f,n) = false
     then
        return false
   EndIf
   For i← 1, m-1 execute
       For j← i+1, m execute
          If  f[i] = f[j]
            then
                  return false
          EndIf
       EndFor
   EndFor
   For i← 1,n execute
      If  exista (m,f,i) = false
         then
                  return false
      EndIf
   EndFor
   return true
EndAlgorithm

Algorithm esteFunctie(m, f, n):
    For i← 1,m   execute
      If (f[i] < 1) OR (f[i] > n)
        then
            return false
      EndIf
    EndFor
    return true
EndAlgorithm

Algorithm exista(m, f, val):
    For i← 1,m   execute
       If  f[i] = val
          then
             return true
       EndIf
    EndFor
    return false
EndAlgorithm
```

**A. true** dacă **f** este funcţie bijectivă;
**B. true** dacă **f** este funcţie surjectivă;
**C. true** dacă **f** este functie;
**D. true** dacă **f** este funcţie injectivă.

**2.31.** Ce se obţine în vectorul **x,** după execuţia algoritmului **pm.** Date: **x** şi **n=lungimea** lui **x.**

```
Algorithm pm(x, n):
  k    ← 1
  p[k] ← 1              {vectorul p este intern algoritmului}
  For i← 2,n execute
    If x[i] > x[p[1]]
       then
             k    ← 1
             p[k] ← i
       else
         If  x[i] = x[p[1]]
           then
                  k    ← k+1
                  p[k] ← i
         EndIf
    EndIf
  EndFor
  j← p[k]
  For i← j-1,1,-1 execute
        x[i]← x[i+1]
  EndFor
EndAlgorithm
```

**A.** Pentru **x= {1, 2, 3, 4, 5, 8, 7, 6}, n=8** la final **x= {7, 7, 7, 7, 7, 7, 7, 7}**
**B.** Pentru **x= {1, 2, 3, 4, 5, 6, 7, 3}, n=8** la final **x= {7, 7, 7, 7, 7, 7, 7, 7}**
**C.** Pentru **x= {1, 2, 8, 4, 5, 8, 7, 6}, n=8** la final **x= {8, 8, 8, 8, 8, 8, 7, 6}**
**D.** Pentru **x= {1, 2, 3, 9, 5, 6, 9, 3}, n=8** la final **x= {1, 2, 3, 5, 6, 3, 9, 9}**

**2.32.** Avem algoritmul **f(...)** cu **4** parametri: **2** vectori **p**, **q**, şi lungimile lor **m,n**.

```
Algorithm f(m,p,n,q):
    k← m+n
    For i← 0,k execute   {r vector intern, k lungimea lui}
           r[i]← 0
    EndFor
    For i← 0,m execute
      For j← 0,n execute
           r[i+j] ← r[i+j] + p[i] * q[j]
      EndFor
    EndFor
EndAlgorithm
```

Dacă **m=3, p= {1, 2, 3, 4}, n=2, q= {1, 2, 3},** care propoziţii sunt adevărate ?

**A. k=3** şi **r= {10, 20, 30, 40};**
**B. k=4** şi **r= {12, 13, 12, 10, 9};**
**C. k=5** şi **r= {1, 4, 10, 16, 17, 12};**
**D. k=5** şi **r= {1, 4, 10, 16, 18, 12}.**

**2.33.** Ce se obţine în vectorul **P** după execuţia algoritmului **pm(...)**?

```
Algorithm PM(n, x):
                        {n e lungimea lui x, k lungimea lui P}
   k← 1
   p[k]← 1
   For i← 2,n execute
     If  x[i] < x[p[1]]
       then
          k   ← 1
          p[k]← i
       else
         If  x[i] = x[p[1]]
           then
             k← k+1
             p[k]← i
          EndIf
     EndIf
   EndFor
EndAlgorithm
```

**A.** Pentru **x= {9, 8, 7, 6, 5, 4, 3, 2, 1}** ⇒ **k=1** şi **P= {1} ;**
**B.** Pentru **x= {1, 3, 2, 4, 6, 5, 8, 9, 9}** ⇒ **k=2** şi **P= {8, 9};**
**C.** Pentru **x= {4, 8, 8, 5, 8, 3, 8, 5, 3}** ⇒ **k=2** şi **P= {6, 9};**
**D.** Pentru **x= {9, 4, 2, 2, 9, 2, 5, 9, 3}** ⇒ **k=3** şi **P= {3, 4, 6}.**

**2.34.** Se dă algoritmul **a(...)** cu **4** parametri între care **x**, **y** sunt vectori şi **m, n** sunt lungimile lor. Trebuie determinaţi **b**, **d** şi **c**. Care afirmaţii sunt corecte?

```
Algorithm a(m,x,n,y):
  b← c← d← 0
  For i← 1,m execute
    For j← 1,n execute
      k← 0
      While (i+k ≤ m) AND (j+k ≤ n) AND (x[i+k]=y[j+k]) execute
                k← k+1
      EndWhile
      If k > d
         then
             b← i
             d← k
             c← j
      EndIf
    EndFor
  EndFor
EndAlgorithm
```

**A.** **m=8, x= {1, 4, -1, 5, 10, 2, 30, 17}**
**n=14, y= {-2, -7, 1, 5, -1, 7, 10, 2, 30, 25, 17, 5, 10, 2}** **⇒ b=5, d=3, c=7;**

**B.** **m=8, x= {1, 4, -1, 5, 10, 2, 30, 17}**
**n=14, y= {-2, -7, 1, 5, -1, 7, 10, 2, 30, 25, 17, 5, 10, 2};** **⇒ b=4, d=3, c=12;**

**C.** **m=5, x= {1, 1, 1, 1, 1}**
**n=7, y= {1, 1, 1, 1, 1, 1, 1}** **⇒ b=1, d=5, c=1;**

**D.** **m=5, x= {1, 1, 1, 1, 1}**
**n=7, y= {1, 1, 1, 1, 1, 1, 1}** **⇒ b=1, d=5, c=2;**

**2.35.** Ce returnează algoritmul **func**. Vectorul **f** are valorile unei funcţii definite pe **{1..m}** cu valori in **{1..n}**.
Exemplu: o funcţie: **{1, 2, 3}** $\rightarrow$ **{1, 2, 3, 4}** este reprezentată de vectorul **f = {2, 4, 3}**

```
Algorithm func (m, f, n):
   If m > n
     then
        returnează false
   EndIf
   If esteFunctie(m,f,n) = false
     then
        return false
   EndIf
   For i← 1, m-1 execute
       For j← i+1, m execute
          If  f[i] = f[j]
            then
               return false
          EndIf
       EndFor
   EndFor
   return true
SfAlgoritm

Algorithm esteFunctie(m, f, n):
    For i← 1,m   execute
      If (f[i] < 1) OR (f[i] > n)
        then
           return false
      EndIf
    EndFor
    return true
EndAlgorithm
```

**A. true** dacă **f** nu este funcţie bijectivă;
**B. true** dacă **f** este funcţie surjectivă;
**C. true** dacă **f** este doar functie;
**D. true** dacă **f** este funcţie injectivă.

**2.36.** Ce returnează algoritmul **func**. Vectorul **f** are valorile unei funcţii definite pe **{1..m}** cu valori in **{1..n}**.
Exemplu: o funcţie: **{1, 2, 3}** $\rightarrow$ **{1, 2, 3, 4}** este reprezentată de vectorul **f = {2, 4, 3}**

```
Algorithm func (m, f, n):
   If m < n
      then
         return false
   EndIf
   If esteFunctie(m,f,n) = false
      then
         return false
   EndIf
   For i← 1,n execute
      If  exista (m,f,i) = false
         then
            return false
      EndIf
   EndFor
   return true
SfALgoritm


Algorithm esteFunctie(m, f, n):
    For i← 1,m   execute
      If (f[i] < 1) sau (f[i] > n)
        then
           return false
      EndIf
    EndFor
    return true
EndAlgorithm

Algorithm exista(m, f, val):
    For i← 1,m   execute
       If  f[i] = val
          then
             return true
       EndIf
    EndFor
    return false
EndAlgorithm
```

**A. true** dacă **f** nu este funcţie bijectivă;
**B. true** dacă **f** este funcţie surjectivă;
**C. true** dacă **f** este doar functie;
**D. true** dacă **f** este funcţie injectivă.

**2.37.** Se dă algoritmul **a(m,x)**. Parametrul **x** este vector iar **m** este lungimea lui,
Ce valori au **b,c,d** după execuţia algoritmului?

```
Algorithm a(m,x):
  b← c← d← 0
  For i← 1,m-1 execute
    For j← i+1,m  execute
       k← 0
       While (j+k ≤ m) AND (x[i+k] = x[j+k]) execute
            k← k+1
       SfCâTimp
       If  k > d
          then
             b← i
             d← k
             c← j
       EndIf
    EndFor
  EndFor
EndAlgorithm
```

**A.** **m=8, X= {1, 4, -1, 5, 10, 2, 30, 17}** ⇒ **b=0,d=0, c=0;**
**B.** **m=8, X= {1, 4, -1, 5, 10, 1, 4, 17}** ⇒ **b=1,d=2, c=6;**
**C.** **m=5, X= {1, 1, 1, 1, 1}** ⇒ **b=1,d=4, c=2;**
**D.** **m=5, X= {1, 1, 1, 1, 1}** ⇒ **b=1,d=5, c=1;**

**2.38.** Se dă algortimul **a(n,x,m,y)** cu **n>0**, **m>0** valori naturale iar **x**,**y** sunt mulţimi de întregi. Algoritmul **a(...)** apelează algoritmul **e(...).** Care afirmaţii sunt corecte?

```
Algorithm a(n,x,m,y):
      k← 0
      For i← 1,n execute
           If e(m,y,x[i]) = 1
              then
                k   ← k+1
                z[k]← x[i]
           EndIf
      EndFor
EndAlgorithm

Algorithm e(n,b,val):
    For i← 1,n execute
        If  val = b[i]
          then
            return 1
        EndIf
    EndFor
    return 0
EndAlgorithm
```

**A.** Dacă **X= {1, 3, 5, 7, 9}; Y= {1, 2, 3, 4, 5}** ⇒ **Z= {1, 2, 3, 4, 5, 5, 7, 9};**
**B.** Dacă **X= {3, 7, 9, 5, 1}; Y= {1, 2, 3, 4, 5}** ⇒ **Z= {3, 5, 1};**
**C.** Dacă **X= {1, 3, 5, 7, 9}; Y= {1, 2, 3, 4, 5}** ⇒ **Z= {7, 9};**
**D.** Dacă **X= {1, 3, 4, 5}; Y= {4, 5}** ⇒ **Z= {4 ,5}.**

**2.39.** Se dă algortimul **b(n,x,m,y)** cu **n>0**, **m>0** valori naturale iar **x**,**y** sunt mulţimi de întregi (vectori). Algoritmul **b(...)** apelează algoritmul **e(...).** Care afirmaţii sunt corecte?

```
Algorithm b(n,x,,y):
      k← 0
      For i← 1,n execute
             k← k+1
           z[k]← x[i]
      EndFor
      For i← 1,m execute
          If  e(n,x,y[i]) = 0
            then
               k   ← k+1
               x[k]← y[i]
           EndIf
       EndFor
 SfAgoritm

Algorithm e(n,x,val):
    For i← 1,n execute
        If  val = x[i]
          then
            return 1
        EndIf
    EndFor
    return 0
EndAlgorithm
```

**A.** Dacă **x= {3, 5, 1, 7, 9}; y= {1, 4, 3, 2, 5}** ⇒ **z= {3, 5, 1, 7, 9, 4, 2};**
**B.** Dacă **x= {1, 3, 5, 7, 9}; y= {1, 2, 3, 4, 5}** ⇒ **z= {1, 3, 5};**
**C.** Dacă **x= {1, 3, 5, 7, 9}; y= {1, 2, 3, 4, 5}** ⇒ **z= {7, 9};**
**D.** Dacă **x= {1, 3, 4, 5}; y= {4, 5}** ⇒ **z= {1, 3, 5}.**

**2.40.** Se dă algortimul **c(n,x,m,y)** cu **n>0**, **m>0** valori naturale iar **x**,**y** sunt mulţimi de întregi (vectori). Algoritmul **c(...)** apelează algoritmul **e(...).** Care afirmaţii sunt corecte?

```
Algorithm c(n,x,m,y):
      k← 0
      For i←1,n execute
            If  e(m,y,x[i]) = 0
               then
                     k   ← k+1
                     z[k]← x[i]
            EndIf
      EndFor
EndAlgorithm

Algorithm e(m,y,val):
    For i← 1,m execute
       If  val = y[i]
         then
           return 1
       EndIf
    EndFor
    return 0
EndAlgorithm
```

**A.** Dacă **x= {9, 3, 7, 5, 1}; y= {1, 2, 3, 4, 5}** ⇒ **z= {1, 2, 3, 4, 5, 7, 9};**

**B.** Dacă **x= {3, 5, 7, 9, 1}; y= {1, 2, 3, 4, 5}** ⇒ **z= {3, 5, 1};**

**C. Dacă x= {7, 1, 5, 9, 3}; y= {4, 2, 1, 5, 3}** ⇒ **z= {7, 9};**

**D.** Dacă **x= {1, 3, 5}; y= {1, 3, 4, 5}** ⇒ **z= {4};**

**2.41.** Algoritmul **creV**(n), crează vectorul **v** cu numere naturale, **n** este dată de intrare.

```
Algorithm creV(n):
    v[1]← m← k← 1
    While  k < n execute
        m← m+1
        i← 1
        While (k < n) AND (i ≤ m) execute
           k← k+1
           v[k]← i
           i← i+1
        SfCătTimp
    SfCătTimp
EndAlgorithm
```

Ce valori are vectorul **v**?

**A.** dacă **n=10** atunci **v= {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}**
**B.** dacă **n=10** atunci **v= {1, 2, 1, 2, 3, 1, 2, 3, 4, 1}**
**C.** dacă **n=11** atunci **v= {1, 1, 1, 2, 1, 2, 3, 1, 2, 3, 4}**
**D.** dacă **n=10** atunci **v= {1, 1, 2, 1, 2, 3, 1, 2, 3, 4}**

**2.42.** Algoritmul **rez(n)**, crează vectorul **v** cu numere naturale, **n** este dată de intrare. Algoritmul **rez(n)** apelează **p(k)**

```
Algorithm rez(n):
    k←3
    nr←0
    While (nr<n) execute
       If (p(k) = 1) AND (p(k+2) = 1)
         then
            nr← nr+1
            v[nr]← k
            nr← nr+1
            v[nr]← k+2
        EndIf
        k← k+2
    EndWhile
EndAlgorithm

Algoritmul p(n):
   If  n < 2
      then
         return 0
   EndIf
   If (n > 2) AND (n MOD 2 = 0)
      then
         return 0
   EndIf
   d← 3
   CâTimp d*d ≤ n execute
       If (n MOD d) = 0)
            then
                  return 0
      EndIf
      d← d+2
   EndWhile
   return 1
EndAlgorithm
```

Ce conţine vectorul **v**?

**A. n=8 ⇒ v= {1, 2, 3, 5, 7, 11, 13, 17}**
**B. n=8 ⇒ v= {3, 5, 7, 11, 13, 17, 19, 23}**
**C. n=8 ⇒ v= {3, 5, 5, 7, 11, 13, 17, 19}**
**D. n=8 ⇒ v= {3, 5, 7, 11, 13, 17, 19, 21}**

**2.43.** Algoritmul **rezolv(n)**, crează vectorul **x** cu numere naturale, **n** este data de intrare; **rezolv()** apelează **prim(k)** care determină primalitatea unui numar natural (1, dacă **k** e prim).

```
Algorithm rezolv(n):
    x[1]← 1
    i← k← 2
    While i ≤ n execute
       If  prim(k) = 1
          then
             j← 1
             While (i ≤ n) AND (j ≤ k) execute
                x[i]← j
                i← i+1
                j← j+1
             EndWhile
          else
             j← 1
             While  (i ≤n) AND (j ≤k)
                    x[i]← k
                    i← i+1
                    j← j+1
             EndWhile
       EndIf
       k← k+1
    EndWhile
EndAlgorithm
```

Ce conţine vectorul **x**?

**A. n=12 ⇒ x= {1, 2, 3, 1, 2, 3, 4, 4, 4, 4, 1, 2};**
**B. n=12 ⇒ x= {1, 1, 2, 1, 2, 1, 2, 3, 4, 4, 4, 4};**
**C. n=12 ⇒ x= {1, 1, 2, 1, 2, 3, 4, 4, 4, 4, 1, 2};**
**D. n=12 ⇒ x= {1, 1, 2, 1, 2, 3, 3, 3, 4, 4, 4, 4}.**

**2.44.** Algoritmul **comp(n,f,g)** compune doi vectori ce reprezinta **2** funcţii, **f** şi **g**. Ce conţine vectorul **gf**?
(**n** lungimea lui **f**, precum şi vectorii **f** şi **g** sunt date).

```
Algoritmul comp(n, f, g):
   For i← 1,n execute
        gf[i] ← g[f[i]]
   EndFor
EndAlgorithm
```

**A. n=4, f= {2, 1, 3, 3}, g= {2, 1, 4} ⇒ gf= {1, 2, 3, 4}**
**B. n=5, f= {5, 1, 2, 3, 4}, g= {1, 1, 1, 2, 3} ⇒ gf= {4, 1, 1, 2, 3}**
**C. n=4, f= {2, 1, 3, 3}, g= {2, 1, 4} ⇒ gf= {1, 2, 4, 4}**
**D. n=5, f= {5, 1, 2, 3, 4}, g={1, 1, 1, 2, 3} ⇒ gf= {3, 1, 1, 1, 2}**

**2.45.** Algoritmul **creV(n)**, crează vectorul **v** cu numere naturale, **n** este data de intrare.

```
Algorithm creV (n):
    v[1]← 1
    i← k← 2
    While  i ≤ n execute
        j← 2
        v[i]← k;
        i← i+1
        While (i ≤ n) AND (j ≤ k DIV 2)
                If (k MOD j) = 0
                  then
                    v[i]← j;
                    i← i+1
                EndIf
                j← j+1
        SfCâTimp
        k← k+1
    EndWhile
EndAlgorithm
```

Care din afirmaţii sunt corecte?

**A. n=12 ⇒ v= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}**
**B. n=14 ⇒ v= {1, 1, 2, 1, 3, 1, 2, 4, 1, 5, 1, 2, 3, 6}**
**C. n=13 ⇒ v= {1, 2, 3, 2, 4, 5, 2, 3, 6, 7, 2, 4, 8}**
**D. n=15 ⇒ v= {1, 2, 3, 4, 2, 5, 6, 2, 3, 7, 8, 2, 4, 9, 3}**

**2.46.** Algoritmul **creV(n)**, crează vectorul **v** cu numere naturale, **n** este dată de intrare.

```
Algorithm creV(n):
    v[1]← 1
    i← k← 2                        ///i indice pentru V
    While i ≤ n execute
        j← 2
        v[i] ← k
        i← i+1
        u← 0
        While (i ≤n) AND (j ≤ (k DIV 2))execute
                If  (k MOD j)=0
                   then
                      v[i] ← j
                      i← i+1
                      If u = 0
                        then
                           u ← k DIV j
                      EndIf
                EndIf
                j← j+1
        EndWhile
        j← 2
        While (i ≤ n) AND (j ≤ U) execute
            v[i]← u
            i← i+1
            j← j+1
        EndWhile
        k← k+1
    EndWhile
EndAlgorithm
```

Care din afirmaţii sunt corecte?

**A. n=12 ⇒ v= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}**
**B. n=14 ⇒ v= {1, 1, 2, 1, 3, 1, 2, 4, 1, 5, 1, 2, 3,6}**
**C. n=16 ⇒ v= {1, 2, 3, 4, 2, 5, 6, 2, 3, 3, 7, 8, 2, 4, 4, 4}**
**D. n=19 ⇒ v= {1, 2, 3, 4, 2, 2, 5, 6, 2, 3, 3, 3, 7, 8, 2, 4, 4, 4, 4}**

**2.47.** Algoritmul **creV(n)**, crează vectorul **v** cu numere naturale, **n** este data de intrare.

```
Algorithm creV (n):
    v[1]← 1
    i← 2
    k← 2
    While  i ≤ n execute
        j← 2
        If  P(k) = 1
           then
              v[i]← k
              i← i+1
           EndIf
        While (i ≤ n) AND (j ≤ (k DIV 2)) execute
                If (k MOD j) = 0
                   then
                      v[i]← j
                      i← i+1
                EndIf
                j← j+1
        EndWhile
        k← k+1
    EndWhile
EndAlgorithm


Algorithm p(n):
   If  n < 2
      then
         return 0
   EndIf
   If (n > 2) AND (n MOD 2 = 0)
      then
         return 0
   EndIf
   d← 3
   CâTimp d*d ≤ n execute
       If (n MOD d) = 0)
            then
                  return 0
      EndIf
      d← d+2
   EndWhile
   return 1
EndAlgorithm
```

Care din afirmaţii sunt corecte?

**A. n=12 ⇒ v= {1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 5}**
**B. n=14 ⇒ v= {1, 2, 3, 2, 5, 2, 3, 7, 1, 4, 3, 2, 3, 6}**
**C. n=16 ⇒ v= {1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, 2, 3}**
**D. n=20 ⇒ v= {1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, 2, 3, 4, 6, 13, 2}**

**2.48.** Fie algoritmul **in(m,x,n,y)**, cu **m,n > 1** şi **x**, **y** vectori ca date de intrare. Algoritmul crează vectorul **z.** Care răspunsuri sunt corecte?

```
Algorithm in(m,x,n,y):
  i← j← 1
  k← 0
  While (i ≤ m) AND (j ≤ n) execute
    If x[i] < y[j]
       then
            k ← k+1
            z[k] ← x[i]
            i ← i+1
       else
           If x[i] > y[j]
              then
                 k ← k+1
                 z[k] ← y[j]
                 j ← j+1
              else
                 i ← i+1
                 j ← j+1
           EndIf
  EndWhile
  While i ≤ m    execute
      k ← k+1
      z[k] ← x[i]
      i ← i+1
  EndWhile
  While  j ≤n execute
      k ← k+1
      z[k] ← y[j]
      j ← j+1
  EndWhile
EndAlgorithm
```

**A.** **m=11, x= {-1, 2, 3, 41, 51 ,61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 41, 41, 51 ,61, 71, 81, 86, 87, 88, 91, 91, 92, 301, 401, 401}**

**B.** **m=11, x= {-1, 2, 3, 41, 51, 61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 41, 41, 51, 61, 71, 81, 86, 87, 88, 91, 91, 92, 301, 401}**

**C.** **m=11, x= {-1, 2, 3, 41, 51, 61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 21, 51, 61, 71, 81, 86, 87, 88, 92, 301}**

**D.** **m=11, x= {-1, 2, 3, 41, 51, 61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 41, 51, 61, 71, 81, 86, 87, 88, 91, 92, 301, 401}**

**2.49.** Fie algoritmul **in(m,x,n,y)**, cu **m,n > 1** şi **x**, **y** vectori ca date de intrare. Algoritmul crează vectorul **z.** Care răspunsuri sunt corecte?

```
Algorithm in(m,x,n,y):
   i← j← 1
   k← 0
   While (i ≤ m) AND (j ≤ n) execute
     If x[i] < y[j]
        then
            k ← k+1
            z[k] ← x[i]
            i ← i+1
        else
            If x[i] > y[j]
               then
                  k ← k+1
                  z[k] ← y[j]
                  j ← j+1
               else
                  k ← k+1
                  z[k] ← x[i]
                  i ← i+1
                  j ← j+1
            EndIf
   EndWhile
   While i ≤ m    execute
       k ← k+1
       z[k] ← x[i]
       i ← i+1
   EndWhile
   While  j ≤n execute
       k ← k+1
       z[k] ← y[j]
       j ← j+1
   EndWhile
EndAlgorithm
```

**A.** **m=11, x= {-1, 2, 3, 41, 51, 61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 21, 41, 41, 51, 61, 71, 81,86,87,88,91,91,92,301,401,401}**

**B.** **m=11, x= {-1, 2, 3, 41, 51, 61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 21, 41, 41, 51, 61, 71, 81, 86, 87, 88, 91, 91, 92, 301, 401}**

**C.** **m=11, x= {-1, 2, 3, 41, 51, 61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 21, 41, 51, 61, 71, 81, 86, 87, 88, 91, 92, 301, 401, 401}**

**D.** **m=11, x= {-1, 2, 3, 41, 51, 61, 71, 81, 91, 301, 401}; n=9, y= {1, 21, 41, 86, 87, 88, 91, 92, 401};** atunci **z= {-1, 1, 2, 3, 21, 41, 51, 61, 71, 81, 86, 87, 88, 91, 92, 301, 401}**

**2.50.** Ce conţine vectorul **z** dupa execuţia algoritmului, pentru vectorii:

**x= {1, 2, 3, 55, 66, 77, 78};**
**y= {1, 2, 44, 57, 66, 77, 79, 200}.**

```
Algorithm a(m,x,n,y):
    i← j← 1
    k← 0
    While (i ≤ m) AND (j ≤ n) execute
          k ← k+1
          If x[i] ≤ y[j]
             then
                z[k] ← x[i]
                i ← i+1
             else
                z[k] ← y[j]
                j ← j+1
            EndIf
    EndWhile
    While i ≤ m      execute
          k ← k+1
          z[k] ← x[i]
          i ← i+1
    EndWhile
    While  j ≤n execute
          k ← k+1
          z[k] ← y[j]
          j ← j+1
    EndWhile
EndAlgorithm
```

**A. z= {1, 1, 2, 3, 44, 55, 66, 77, 78, 79, 200}**
**B. z= {1, 1, 2, 2, 44, 55, 57, 66, 66, 77, 77, 78, 79, 200}**
**C. z= {1, 1, 2, 2, 3, 44, 55, 57, 66, 66, 77, 77, 78, 79, 200}**
**D. z= {1, 1, 2, 2, 3, 44, 55, 66, 77, 78, 79, 200}**

**2.51.** Ce conţine vectorul **z** după execuţia algoritmului, pentru vectorii:

**x= {2, 3, 41, 51, 61, 401};**
**y= {1, 2, 41, 86, 87, 91, 92, 401}.**

```
Algorithm a(m,x,n,y):
   i← j← 1
   k← 0
   While (i ≤ m) AND (j ≤ n) execute
         If x[i] ≤ y[j]
            then
               k ← k+1
               z[k] ← x[i]
               i ← i+1
            else
               k ← k+1
               z[k] ← y[j]
               j ← j+1
           EndIf
   EndWhile
   While i ≤ m      execute
         k ← k+1
         z[k] ← x[i]
         i ← i+1
   EndWhile
   While  j ≤n      execute
         k ← k+1
         z[k] ← y[j]
         j ← j+1
   EndWhile
EndAlgorithm
```

**A. z= {1, 3, 51, 62, 86, 87, 91, 92}**
**B. z= {1, 2, 3, 41, 51, 61, 86, 87, 91, 92, 401}**
**C. z= {1, 2, 3, 41, 41, 51, 61, 86, 87, 92, 401, 401}**
**D. z= {1, 2, 2, 3, 41, 41, 51, 61, 86, 87, 91, 92, 401, 401}**

**2.52.** Ce conţine vectorul **v** după execuţia algoritmului **creV(n)**, **n** natural, **n>1**?

```
Algorithm creV(n):
        v[1]← m← k← 1
        While  k < n execute
              m← m+1
              i← 1
               While (k < n) AND (i ≤ m) execute
                    k← k+1
                    v[k]← i
                    i ← i+1
              EndWhile
        EndWhile
EndAlgorithm
```

**A.** pentru **n=10, => v= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}**
**B.** pentru **n=10, => v= {1, 2, 1, 3, 2, 1, 4, 3, 2, 1}**
**C.** pentru **n=12, => v= {1, 1, 2, 1, 2, 3, 1, 2, 3, 4}**
**D.** pentru **n=11, => v= {1 ,1, 2, 1, 2, 3, 1, 2, 3, 4, 1}**

**2.53.** Ce conţine vectorul **v** după execuţia algoritmului **creV(n)**, **n** natural, **n>1**?

```
Algorithm CreV (n):
      v[1]← 1
      i← 2
      k← 2
      While i ≤ n execute
            j← 2
            v[i]← k
            i ← i+1
            While (i ≤ n) AND (j ≤ (k DIV 2)) execute
                  If  (k MOD j) = 0
                     then
                         v[i]← j
                         i ← i+1
                  EndIf
                  j ← j+1
            EndWhile
            k ← k+1
      EndWhile
EndAlgorithm
```

**A.** pentru **n=15 ⇒ v= {1, 2, 3, 4, 2, 5, 6, 2, 3, 7, 8, 2, 4, 9, 3}**
**B.** pentru **n=15 ⇒ v= {1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, 2}**
**C.** pentru **n=12 ⇒ v= {1, 2, 3, 4, 2, 5, 6, 2, 3, 7, 2, 4}**
**D.** pentru **n=12 ⇒ v= {1, 2, 3, 4, 2, 5, 6, 2, 3, 7, 8, 2}**

**2.54.** Ce conţine vectorul **v** după execuţia algoritmului **creV(n)**, **n** natural, **n>1**?

```
Algorithm creV(n):
      i← k← 0
      While  i < n execute
            k← k+1
            j← 1
            While (i < n) AND (j ≤ k)  execute
                  i← i+1
                  v[i] ← j
                  j← j+1
             EndWhile
      EndWhile
EndAlgorithm
```

**A. n=10 ⇒ v= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}**
**B. n=12 ⇒ v= {1, 1, 2, 1, 2, 3, 1, 2, 3, 4}**
**C. n=13 ⇒ v= {1, 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3}**
**D. n=14 ⇒ v= {1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2}**

**2.55.** Se dă algoritmul **a(m,x,n,y)**. Care afirmaţii sunt adevărate?

```
Algorithm a(m,x,n,y):
  b← c← d← 0
  For i← 1,m execute
    For j← 1,n execute
      k← 0
      While (i+k ≤ m) AND (j+k ≤ n) AND (x[i+k] = y[j+k]) execute
                  k← k+1     (*)
      EndWhile
      If k > d
         then
             b← i           (**)
             d← k
             c← j
      EndIf
    EndFor
  EndFor
EndAlgorithm
```

**A.** Grupul de **3** instrucţiuni care începe la **(**)** se execute de **4** ori pentru
**m=8, x= {1, 4, -1, 5, 10, 2, 30, 17}; n=12, y= {-2, -7, 1, 5, -1, 7, 10, 2, 30, 17, 5, 10};**

**B.** Grupul de **3** instrucţiuni care începe la **(**)** se execute de **3** ori pentru
**m=8, x= {1, 4, -1, 5, 10, 2, 30, 17}; n=12, y={-2, -7, 1, 5, -1, 7, 10, 2, 30, 17, 5, 10};**

**C.** Instrucţiunea **(*)**, adică `k← k+1` se execute de **8** ori pentru
**m=7, x= {1, 4, -1, 5, 10, 2, 30}; n=10, y= {-2, -7, 1, 5, -1, 7, 10, 2, 30, 17};**

**D.** Instrucţiunea **(*)**, adică `k← k+1` se execute de **9** ori pentru
**m=7, x= {1, 4, -1, 5, 10, 2, 30}; n=10, y= {-2, -7, 1, 5, -1, 7, 10, 2, 30, 17};**

**2.56.** Se dă algoritmul **a(m,x)**. (**x** este vector, iar **m** lungimea lui **x**). Care afirmaţii sunt adevărate?

```
Algorithm a(m,x):
  d← 0
  For i← 1,m-1 execute
    For j← i+1,m  execute
       k← 0
       While (j+k ≤ m) AND (x[i+k] = x[j+k]) execute
            k← k+1          (*)
       SfCâTimp
       If  k > d
          then
             d← k           (**)
       EndIf
    EndFor
  EndFor
EndAlgorithm
```

**A.** **m=10, x= {1, 4, -1, 5, 10, 1, 4, 17, -1, 5};** instrucţiunea **(**)** se execute de **2** ori;
**B.** **m=10, x= {1, 4, -1, 5, 10, 1, 4, 17, -1, 5};** instrucţiunea **(*)** se execute **6** ori;
**C.** **m=4, x= {1, 1, 1, 1};** instructiunea **(**)** se execute o dată;
**D.** **m=4, x= {1, 1, 1, 1};** instrucţiunea **(*)** se execute de **10** ori.

**2.57.** Se dă algoritmul **a(n,x,m,y,k,z)** cu **n>0**, **m>0**, **k** valori naturale iar **x**,**y**,**z** sunt şiruri de numere naturale.
Algoritmul **a(...)** apelează algoritmul **e(m,y,val)** cu **n>0** şi **val** naturale, precum şi **y** vector.
Care afirmaţii sunt adevărate? (Algoritmul **a(...)** este apelat cu **a(n,x,m,y,0,z))**

```
Algorithm a(n,x,m,y,k,z):
       If  n > 0
          then
             If e(m,y,x[n]) = 1
               then
                  k← k+1
                  z[k]=x[n]
             EndIf
             a(n-1,x,m,y,k,z)
       EndIf
EndAlgorithm

Algorithm e(m,y,val):
       For i← m,1,-1 execute
            If  val = t[i]
                then
                   return 1
            EndIf
       EndFor
       return 0
EndAlgorithm
```

**A.** Dacă **x= {1, 3, 5, 7, 9}, n=5; y= {1, 2, 3, 4, 5}, m=5 ⇒ Z= {1, 2, 3, 4, 5, 5, 7, 9};**
**B.** Dacă **x= {3, 7, 9, 5, 1}, n=5; y= {1, 2, 3, 4, 5}, m=5 ⇒ Z= {1, 5, 3};**
**C.** Dacă **x= {1, 3, 5, 7, 9}, n=5; y= {1, 2, 3, 4, 5}, m=5 ⇒ Z= {3, 5, 1};**
**D.** Dacă **x= {1, 3, 5, 7, 9}, n=5; y= {1, 2, 3, 4, 5}, m=5 ⇒ Z= {5, 3, 1}.**

**2.58.** Ce face algoritmul următor? (**n**,**k** naturale, **x**, **p** tablouri de întregi, **x**, **n** date de intrare; **k**, **p** trebuie determinate.

```
Algorithm pmn(n,x):
    i← 1
   While (i < n) AND (x[i] ≥ 0)
       i← i+1
   EndWhile
   If i ≤ n
       then
          k← 1
          p[k]← i;
          For j← i+1,n execute
              If  x[j] = x[p[1]]
                  then
                     k← k+1
                     p[k] ← j
                  else
                     If  (x[j] < 0) AND (x[j] > x[p[1]])
                        then
                           k← 1
                           p[k]← j
                     EndIf
             EndIf
          EndFor
       else
          k← 0
   EndIf
EndAlgorithm
```

**A. n= 9, x= { 2, -5, 3, -2, -9, 1, 3, -2, 1} ⇒ k=2 şi p= {4, 8};**
**B. n= 8; x= {-4, 2, 5, 1, -5, 1, 3, 4} ⇒ k=1 şi p={5};**
**C. n=10; x= { 2, 3, 4, 65, 31, 2, 4, 5, 45, 10} ⇒ k=1 şi p={4};**
**D. n=10; x= { 2, 3, 4, 65, 31, 2, 4, 5, 45,10} ⇒ k=0.**

**2.59.** Ce conţine vectorul **y** (**n** natural, lungimea vectorilor **x** şi **y**) dupa execuţia algorimului **creareY(n,x,y)?** Algoritmul **creareY(...)** apelează **mm(n,x,i)**.

```
Algorithm CreareY(n,x,y):
  If n > 0
      then
         y[n]← mm(n,x,n)
         creareY(n-1,x,y)
  EndIf
EndAlgorithm

Algorithm mm(n,x,i):
      j← 1
      cnt← 0
      While j < i execute
          If x[j] < x[i]
             then
                 cnt← cnt+1
          EndIf
          j← j+1
      EndWhile
      return cnt
 EndAlgorithm
```

**A.** **x= {1, 2, 3, 4, 5, 6} ⇒ y= {0, 1, 2, 3, 4, 5};**
**B.** **x= {1, 2, 3, 4, 5, 6} ⇒ y= {5, 4, 3, 2, 1, 0};**
**C.** **x= {1, 20, 3, 40, 5, 6} ⇒ y= {0, 1, 1, 3, 2, 3};**
**D.** **x= {1, 2, 3, 4, 5, 6} ⇒ y= {1, 1, 1, 1, 1, 1}.**

**2.60.** Algoritmul **creY(...)** apelează, celelalte **2** metode; **m,n>0** naturale; vectorul **x** şi lungimea **n**, sunt date; trebuie creat vectorul **y** şi lungimea **m**.

```
Algorithm creY(n, x, m, y):
     m← 0
     For i← 1,n execute
        poz← existaInY(m,y,x[i])
        If poz = 0
          then
             m← adaugInY(m,y,x[i])
        EndIf
     EndFor
 EndAlgorithm

 Algorithm existaInY(m, y, v)
    For i← 1,m execute
       If v = y[i]
          then
             returnează i
        EndIf
    EndFor
    returnează 0
 EndAlgorithm

 Algorithm adaugInY (m, y, v)
    j← m
    CatTimp (j > 0) AND  (v > Y[j]) execute
        y[j+1]← y[j]
        j     ← j-1
    EndWhile
    y[j+1] ← v
    m← m+1
    return m
 EndAlgorithm
```

**A.** Dacă **x= {1, 2, 3, 4, 5, 1, 2, 3, 4}, n=9** ⇒ **y= {5, 4, 3, 2, 1} şi m=5;**
**B.** Dacă **x= {1, 2, 30, 2, 5, 1, 2, 5, 4}, n=9** ⇒ **y= {1, 2, 4, 5, 30} şi m=5;**
**C.** Dacă **x= {1, 1, 1, 1, 1, 1}, n=6** ⇒ **y= {1} şi m=1;**
**D.** Dacă **x= {1, 2, 30, 2, 5, 1, 2, 5, 4}, n=9** ⇒ **y= {1, 2, 30, 5, 4} şi m=5.**

**2.61.** Care afirmaţii sunt adevărate, după execuţia subprogramului **cmlsf (n,x)?**
Parametri sunt: **x** vector şi **n** lungimea sa, **n>2**.

```
Algorithm cmlsf(n,x):
   i← 1
   st← 0
   dr← -1
   While  i < n exexută
       j← i+1
       While (j < n) AND ((X[j]-X[j-1])*(X[j+1]-X[j])<0)
            j← j+1
       EndWhile
       If j ≤ n
          then
              If ((j-i) > (dr-st)) AND (j-i ≥ 2)
                    then
                          st← i
                          dr← j
              EndIf
       EndIf
       i← j
  EndWhile
EndAlgorithm
```

**A.** Pentru **x= {1, 8, 3, 7, 4, 5, 2, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1}, n=19 ⇒ st=1, dr=8;**
**B.** Pentru **x= {1, 8, 3, 7, 4, 5, 2, 1, -10, 1, -5, 4, -1, 15, -11, 1, -1, 1, -1}, n=19 ⇒ st=8, dr=19;**
**C.** Pentru **x= {6, 5, 4, 4, 5, 6, 7}, n=7 ⇒ st=2, dr=6;**
**D.** Pentru **x= {1, 8, 3, 7, 4, 5, 2, 2, 2, 1, -1, 1, -1, 1, -1, 1}, n=16 => st=10, dr =16.**

**2.62.** Ce se returnează după execuţia metodei **v**, indice de la **1**.

```
Algorithm v(n,x):
  cont← 1
  uV1 ← 2
  For i← 3,n execute
    If x[i]*(uV1-1) > (i-1)*x[uV1]
      then
         cont← cont+1
         uV1← i
    EndIf
  EndFor
  return cont
EndAlgorithm
```

**A.** **n=4; x= {0, 3, 6, 6}** ⇒ valoarea **2;**

**B.** **n=4; x= {0, 2, 6, 7}** ⇒ valoarea **1;**

**C.** **n=6; x= {0, 3, 6, 6, 13, 17}** ⇒ valoarea **3;**

**D.** **n=7; x= {0, 3, 6, 6, 13, 17, 18}** ⇒ aceeaşi valoare ca la punctul **C.**

**2.63.** Ce conţine vectorul **a** de lungime **n**, după executarea algoritmului **aran(a,n)**?

```
Algorithm aran(a,n):
  i← j← 1
  While j ≤ n execute
      If  (a[j] MOD 2 = 1)
        then
            j← j+1
        else
            aux  ←a[j]
            a[j] ← a[i]
            a[i] ← aux
            j← j+1
            i← i+1
      EndIf
  EndWhile
EndAlgorithm
```

**A.** **a= {3, 4, 5, -6, -5, 20, -20, 0, 17, -18}, n=10; ⇒ a= {4, -6, 20, -20, -18, 3, -5, 17, 5};**
**B.** **a= {3, 4, 5, -6, -5, 20, -20, 0, 17, -18}, n=10; ⇒ a= {3, 5, -5, 17, 4, 20, -20, 0, -6, -18};**
**C.** **a= {3, 4, 5, -6, -5, 20, -20, 0, 17, -18}, n=10; ⇒ a= {3, 4, 5, -6, -5, 20, -20, 0, 17, -18};**
**D.** **a= {3, 4, 5, -60, -5, 200, -20, 0, 17, -18}, n=10; ⇒ a= {4, -60, 200, -20, 0, -18, 3, -5, 17, 5}.**

**2.64.** Care afirmaţii sunt adevărate după execuţia algoritmului **pm(x,n)**:

```
Algorithm pm(x,n):          {vectorul p se creează}
      k← 1
      p[1]← 1
      For i← 2,n  execute
         If (x[i]>x[p[1]])
           then
                k← 1
                p[1]← i
           else
              If x[i]=x[p[1]]
                 then
                      k← k+1
                      p[k]← i
              EndIf
         EndIf
      EndFor
      val← x[p[k]]
      i← j← 1
      While  j ≤ n  execute
           If x[j]=val
                 then
                      j← j+1
                 else
                      x[j]←→x[i]
                      i←i+1
                      j←j+1
           EndIf
      EndWhile
EndAlgorithm
```

**A.** Pentru **x= {1, 2, 3, 4, 5, 6, 7, 8}, n=8** la final **x= {2, 3, 4, 5, 6, 7, 8, 1}**
**B.** Pentru **x= {1, 2, 3, 4, 5, 6, 7, 3}, n=8** la final **x= {7, 1, 2, 4, 5, 6, 3, 3}**
**C.** Pentru **x= {1, 2, 8, 4, 5, 6, 7, 8}, n=8** la final **x= {1, 2, 8, 4, 5, 6, 7, 8}**
**D.** Pentru **x= {1, 2, 3, 7, 5, 6, 7, 3}, n=8** la final **x= {1, 2, 3, 5, 6, 3, 7, 7}**

**2.65.** Ce returnează următorul algoritm **m(...)?**

```
Algorithm m (s,n):                  {s-sir, n- lungimea sirui}
      i← 1                          {aprioric s este sortat crescător}
      cont← 0
      While  i ≤ n  execute
            j← i + 1
            While (j ≤ n) AND (s[i]=s[j]) execute
                  j← j + 1
            EndWhile
            cont← cont + 1
            i← j
      EndWhile
      return cont
EndAlgorithm
```

**Care afirmaţii sunt adevărate ?**

- **A. m(...)** returnează numărul de secvenţe de lungime **2,** care au termenii egali;
- **B. m(...)** returnează numărul de elemente ale celei mai lungi secvenţe cu termeni egali;
- **C. m(...)** returnează cardinalul mulţimii formate din elementele lui **s**;
- **D. m(...)** returnează **n**, dacă toate elementele din s sunt egale.

**2.66.** Ce returnează următorul algoritm **m(...)?**

```
Algorithm m (s,n):                  {s-sir, n- lungimea sirui}
      i← 1                          {aprioric s este sortat crescător}
      cont← 0
      While  i ≤ n  execute
            j← i + 1
            While (j ≤ n) AND (s[i]=s[j]) execute
                  j← j + 1
                  cont← cont + 1
            EndWhile
            i← j
      EndWhile
      return cont
EndAlgorithm
```

**Care afirmaţii sunt adevărate ?**

- **A. m(...)** returnează numărul de secvenţe care au termenii egali;
- **B. m(...)** returnează numărul de elemente ale celei mai lungi secvenţe cu termeni egali;
- **C. m(...)** returnează cardinalul mulţimii formate din elementele lui s;
- **D. m(...)** returnează **n-1,** dacă toate elementele din s sunt egale.

**2.67.** Algoritmul **fda(...)** schimbă ordinea elementelor iniţiale din vectorul **x** de lungime **n**. Valoarea **a** este dată de intrare ca şi **x** şi **n**. Care afirmaţii sunt adevărate?

```
Algorithm fda(a,x,n):
    i← j← 1
    While j ≤ n execute
       If x[j] ≥ a
          then
             x[j]←→x[i]
             i← i+1
       EndIf
       j← j+1
    SfCâTimp
EndAlgorithm
```

**A.** If **x= {1, 2, 3, -5, 6, -7}, n=6, a=4** ⇒ **x= {6, 2, 3, -5, 1, -7};**
**B.** If **x= {1, 2, 30, 2, 5, 1}, n=6, a=4** ⇒ **x= {30, 5, 1, 2, 2, 1};**
**C.** If **x= {1, 2, 30, -5, 6, -7}, n=6, a=5** ⇒ **x= {6, 1, 2, 30, -5, -7};**
**D.** If **x= {1, 2, 30, 2, 5, 1}, n=6, a=6** ⇒ **x= {30, 2, 1, 2, 5, 1}.**

**2.68.** Algoritmul **ea(...)** schimbă ordinea elementelor iniţiale din vectorul **x** de lungime **n**, şi uneori modifică lungimea lui x. Valoarea **a** este dată de intrare ca şi **x** şi **n**.

```
Algorithm ea(n,x,a):
      i← 1
      While i ≤ n execute
            If x[i] > a
                  then
                    For j← i,n-1  execute
                        x[j]← x[j+1]
                    EndFor
                     n ← n-1
                  else
                    i← i+1
            EndIf
      EndWhile
 SfAlgorithm
```

**A.** Dacă **x= {1, 2, 3, -5, 6, -7}, n=6, a=4** ⇒ **x= {1, 2, 3, -5, -7}, n=5;**
**B.** Dacă **x= {1, 2, 30, 2, 5, 1}, n=6, a=2** ⇒ **x= {30, 5}, n=2;**
**C.** Dacă **x= {1, 2, -30, 2, 5, 1}, n=6, a=-4** ⇒ **x= {-30}, n=1;**
**D.** Dacă **x= {1, 2, 5, 9, 10, 20}, n=6, a=5** ⇒ **X= {1, 2, 5}, n=3.**

**2.69.** Fie şirul **a: -5, 7, -5, 7, ...**
Care afirmaţii sunt adevărate?

**A.** Termenul general este $a_n = 1.5 + 6.5*(-1)^n$;
**B.** Termenul general este $a_n = 1 + 6*(-1)^n$;
**C.** Suma primilor **2*n** termeni este un număr par;
**D.** Suma primilor **2*n-1** termeni este **< 0.**

**2.70.** Subalgoritmul **generare(n)** prelucrează un număr natural **n (0 < n < 100**). (Model FMI)

```
Algorithm generare(n):
      nr← 0
      For i← 1,1801 execute
            folosit_i ← fals
      EndFor
      While folosit_n = fals execute
            suma← 0
            folosit_n ← adevărat
            While n ≠ 0 execute
                  cifra← n MOD 10
                  n← n DIV 10
                  suma← suma + cifra * cifra * cifra
            EndWhile
            n← suma
            nr← nr + 1
      EndWhile
      returnează nr
EndAlgorithm
```

**Precizați care este efectul acestui subalgoritm.**

**A.** Calculează, în mod repetat, suma cuburilor cifrelor numărului **n** până când suma egalează numărul **n** și returnează numărul repetărilor efectuate;
**B.** Calculează suma cuburilor cifrelor numărului **n** și returnează această sumă;
**C.** Calculează suma cuburilor cifrelor numărului **n**, înlocuiește numărul **n** cu suma obținută și returnează această sumă;
**D.** Calculează numărul înlocuirilor lui **n** cu suma cuburilor cifrelor sale până când se obține o valoare calculată anterior sau numărul însuși și returnează numărul de repetări.

**2.71.** Algoritmul de mai jos are ca parametri de intrare un vector ***v*** cu ***n*** numere naturale (***v***[1], ***v***[2], …,***v***[***n***]) și numărul întreg ***n*** (1 ≤ ***n*** ≤ 10000).

```
Algorithm fn(v, n):
      a ← 0
      For i ← 1, n execute
            ok ← True
            b ← v[i]
            While (b ≠ 0) AND (ok = True) execute
                  If b MOD 2 = 1 then
                        ok ← False
                  EndIf
                  b ← b DIV 10
            EndWhile
            If ok = True then
                  a ← a + 1
            EndIf
      EndFor
      return a
EndAlgorithm
```

Precizați care dintre următoarele afirmații sunt **adevărate**:
A. Algoritmul returnează numărul elementelor impare din vectorul ***v***.
B. Algoritmul returnează numărul elementelor din vectorul ***v*** care sunt puteri ale lui 2.
C. Algoritmul returnează numărul elementelor din vectorul ***v*** care au în componența lor doar cifre pare.
D. Algoritmul returnează numărul elementelor din vectorul ***v*** care au în componența lor doar cifre impare.

**2.72.**
**2.73.**
**2.74.**

# 3. Matrice

**3.1.** Se dă o matrice **a** de cel mult **20** de linii şi coloane. Se mai dau **m**-numărul de linii efectiv şi **n**-numărul de coloane efectiv; **m**,**n** din {**2,3,4,5,...,20**}; valorile matricii sunt întregi.
Ce valori au variabilele **lin** şi **col**, după execuţia secvenţei?

```
lin← col← 0
i← 1
While i ≤ m execute
     j← 1
     While j ≤ n execute
        If a[i][j] ≥ 0
            then
                  lin← i
                  col← j
                  i← m+1
                  j← n+1
        EndIf
        j← j+1
     EndWhile
     i← i+1
EndWhile

If lin > 0
    then
        For i← lin,m  execute
             For j← 1,n  execute
                If (a[i][j] ≥ 0) AND (a[i][j] < a[lin][col])
                   then
                        lin← i
                        col← j
                EndIf
             SfPenru
        EndFor
EndIf
```

**A.** **lin şi col** conţin numărul coloanei, respectiv al liniei în care se găseşte valoarea maximă din matrice;
**B.** **lin şi col** conţin numărul liniei, respectiv al coloanei în care se găseşte valoarea maximă din matrice sau **lin=col=0;**
**C.** **lin şi col** conţin numărul liniei, respectiv al coloanei în care se găseşte valoarea minimă pozitivă din matrice sau **lin=col=0;**
**D.** **lin şi col** conţin numărul coloanei, respectiv al liniei în care se găseşte valoarea minimă pozitivă din matrice sau **lin=col=0**.

**3.2.** Algoritmul **creMat (n)** iniţilizează matricea **a[n][n]**, cu **2 ≤ n ≤ 10**. Ce elemente are matricea **a** (linie cu linie)?

```
Algorithm creMat(n):
      k← 0
      For j← 1,n  execute
         For i← 1,j execute
            k← k+1
            a[i][j]← k
         EndFor
         For i← j-1,1,-1 execute
            k← k+1
            a[j][i]← k
         EndFor
      EndFor
EndAlgorithm
```

**A. n =5 ⇒ 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25;**
**B. n =5 ⇒ 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25;**
**C. n =5 ⇒ 1,2,5,10,17 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21;**
**D. n =5 ⇒ 1,2,3,4,5, 16,17,18,19,6, 15,24,25,20,7, 14,23,22,21,8, 13,12,11,10,9.**

**3.3.** Algoritmul **creMat (n)** iniţilizează matricea **a[n][n]**, cu **2 ≤ n ≤ 10**. Ce elemente are matricea **a** (linie cu linie)?

```
Algorithm creMat(n):
      k← 0
      For i← 1,n  execute
         For j← 1,i execute
            k← k+1
            a[i][j]← k
         EndFor
         For j← i-1,1,-1 execute
               k← k+1
               a[j][i]← k
         EndFor
      EndFor
EndAlgorithm
```

**A. n =5 ⇒ 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25;**
**B. n =5 ⇒ 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25;**
**C. n =5 ⇒ 1,2,5,10,17 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21;**
**D. n =5 ⇒ 1,4,9,16,25, 2,3,8,15,24, 5,6,7,14,23, 10,11,12,13,22, 17,18,19,20,21;**

**3.4.** Algoritmul **creMat (n)** iniţilizează matricea **a[n][n]**, cu **2 ≤ n ≤ 10**. Ce elemente are matricea **a** (linie cu linie)?

```
Algorithm creMat(n):
      For i←1,n execute
            For j← 1,n execute
                If  i ≤ j
                  then
                      a[i][j] ← (j-1)*(j-1)+i
                  else
                      a[i][j] ← i*i-j+1
                EndIf
            EndFor
      EndFor
EndAlgorithm
```

**A.** **n =5 ⇒ 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25;**
**B.** **n =5 ⇒ 1,2,5,10,17, 4,3,6,11,18, 9,8,7,12,19, 15,16,13,14,20, 25,24,23,22,21;**
**C.** **n =5 ⇒ 1,2,3,4,5, 16,17,18,19,6, 15,24,25,20,7, 14,23,22,21,8, 13,12,11,10,9;**
**D.** **n =5 ⇒ 1,2,5,10,17, 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21.**

**3.5.** Algoritmul **creMat (n)** iniţilizează matricea **a[n][n]**, cu **2 ≤ n ≤ 10**. Ce elemente are matricea **a** (linie cu linie)?

```
Algorithm creMat(n):
      For i←1,n execute
            For j← 1,n execute
                If  j ≤ i
                  then
                      a[i][j] ← (i-1)*(i-1)+j
                  else
                      a[i][j] ← j*j-i+1
                EndIf
            EndFor
      EndFor
EndAlgorithm
```

**A.** **n =5 ⇒ 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25;**
**B.** **n =5, ⇒ 1,4,9,16,25, 2,3,8,15,24, 7,6,5,14,23, 12,11,10,13,22, 21,19,18,20,17;**
**C.** **n =5, ⇒ 1,2,5,10,17, 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21;**
**D.** **n =5, ⇒ 1,4,9,16,25, 2,3,8,15,24, 5,6,7,14,23, 10,11,12,13,22, 17,18,19,20,21.**

**3.6.** Algoritmul **creMat (n)** iniţilizează matricea **a[n][n]**, cu **2 ≤ n ≤ 10**. Ce elemente are matricea **a** (linie cu linie)?

```
Algorithm creMat(n):
      k←0
      For d←1,n execute
            For j← 1,d execute
                k←k+1
                a[d+1-j][j]← k
            EndFor
      For d←2,n execute
            For j← d,n execute
                k←k+1
                a[n+d-j][j]← k
            EndFor
     EndFor
EndAlgorithm
```

**A. n =5 ⇒ 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.**
**B. n =5 ⇒ 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.**
**C. n =5 ⇒ 1,3,6,10,15, 2,5,9,14,19, 4,8,13,18,22, 7,12,17,21,24, 11,16,20,23,25.**
**D. n =5 ⇒ 1,4,9,16,25, 2,3,8,15,24, 5,6,7,14,23, 10,11,12,13,22, 17,18,19,20,21.**

**3.7.** Algoritmul **creMat (n)** iniţilizează matricea **a[n][n]**, cu **2 ≤ n ≤ 10**. Ce elemente are matricea **a** (linie cu linie)?

```
Algorithm creMat(n):
      k←0
      For d←1,n execute
            For i← 1,d execute
                k←k+1
                a[i][d+1-i]← k
            EndFor
      For d←2,n execute
            For i← d,n execute
                k←k+1
                a[i][n+d-i]← k
            EndFor
     EndFor
EndAlgorithm
```

**A. n =5 ⇒ 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25;**
**B. n =5 ⇒ 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25;**
**C. n =5 ⇒ 1,2,5,10,17, 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21;**
**D. n =5 ⇒ 1,2,4,7,11, 3,5,8,12,16, 6,9,13,17,20, 10,14,18,21,23, 15,19,22,24,25.**

**3.8.** Avem matricea:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 |

Care din algoritmii următori afişează elementele matricii în ordinea (linie după linie)?

**1,2,3,4,5,6, 12,11,10,9,8,7, 13,14,15,16,17,18, 24,23,22,21,20, 25,26,27,28,29,30, 36,35,34,33,32,31**

**A.**

```
Algorithm afisMatrice (n)       {n=6 la apel}
       For i← 1,n execute
          If (i MOD 2) = 0
                then
                      For j← n,1,-1 execute
                         Write a[i][j] + ’,’
                      EndFor
                else
                      For j← 1,n  execute
                         Write a[i][j] + ’,’
                      EndFor
          EndIf
       EndFor
EndAlgorithm
```

**B.**

```
Algorithm afisMatrice (n)       {n=6 la apel}
       For i← 1,n execute
          If (i MOD 2) = 0
                then
                      For j← 1,n execute
                           Write a[i][n-j] + ’,’
                      EndFor
                else
                      For j← 1,n  execute
                           Write a[i][j] + ’,’
                      EndFor
          EndIf
       EndFor
EndAlgorithm
```

**C.**

```
Algorithm afisMatrice (n)       {n=6 la apel}
       For j← 1,n execute
          If (j MOD 2) = 0
                then
                      For i← 1,n execute
                            Write a[j][n-i+1] + ',’
                      EndFor
                else
                      For i← 1,n  execute
                            Write a[j][i] + ',’
                      EndFor
          EndIf
       EndFor
EndAlgorithm
```

**D.**

```
Algorithm afisMatrice (n)       {n=6 la apel}
       For i← 1,n execute
          If (i MOD 2) = 1
                then
                      For j← n,1,-1 execute
                            Write a[i][n-j+1] + ',’
                      EndFor
                else
                      For j← 1,n  execute
                            Write a[i][n-j+1] + ',’
                      EndFor
          EndIf
       EndFor
EndAlgorithm
```

**3.9.** Avem matricea:

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |

Care din metodele următoare afişează elementele matricii în ordinea următoare (paralele cu diagonala principală, inclusiv diagonala principală)?

21, 16,22, 11,17,23, 6,12,18,24, 1,7,13,19,25, 2,8,14,20, 3,9,15, 4,10, 5

**A.**

```
Algorithm a(n):                          {n=5 la apel}
    For  k← 1,3*n execute
       If  k ≤ n
           then
              For  i← 1,k execute
                  Write a[n-k+i][i] + ','
              EndFor
           else
              For  i← k,2*n-1 execute
                  Write  a[i-k+1][i-n+1] + ',';
              EndFor
      EndIf
EndAlgorithm
```

**B.**

```
Algorithm b(n):                          {n=5 la apel}
    For  k← 1,2*n+1 execute
       If  k ≤ n
           then
              For  i← 1,k execute
                  Write a[n-k+i][i] + ','
              EndFor
           else
              For  i← k,2*n-1 execute
                  Write  a[i-k+1][i-n+1] + ',';
              EndFor
      EndIf
EndAlgorithm
```

**C.**

```
Algorithm c(n):                          {n=5 la apel}
    For  k← 1,2*n-1 execute
       If  k ≤ n
           then
              For  i← 1,k execute
                  Write a[n-k+i][i] + ','
              EndFor
           else
              For  i← k,2*n-1 execute
                  Write  a[i-k+1][n-i+1] + ',';
              EndFor
      EndIf
EndAlgorithm
```

**D.**

```
Algorithm d(n):                          {n=5 la apel}
    For  k← 1,2*n-1 execute
       If  k ≤ n
           then
              For  i← 1,k execute
                  Write a[n-k+i][i] + ','
              EndFor
           else
              For  i← k+1,2*n-1 execute
                  Write  a[i-k+1][i-n+1] + ',';
              EndFor
      EndIf
EndAlgorithm
```

**3.10.** Avem matricea:

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |

Care din metodele următoare afişează elementele matricii în ordinea (paralele cu diagonala secundară):

**1, 2,6, 3,7,11, 4,8,12,16, 5,9,13,17,21, 10,14,18,22, 15,19,23 20,24 25**

**A.**

```
Algorithm a(n):                              {n=5 la apel}
    For  k← 1,3*n execute
       If  k ≤  n
          then
             For  i← 1,k execute
                 Write a[i][k+1-i] + ','
             EndFor
          else
             For  i← k,2*n-1 execute
                 Write  a[i-n+1][n+k-i] + ',';
             EndFor
      EndIf
EndAlgorithm
```

**B.**

```
Algorithm b(n):                              {n=5 la apel}
    For  k← 1,3*n-1 execute
       If  k ≤  n
          then
             For  i← 1,k execute
                 Write a[i][k+1-i] + ','
             EndFor
          else
             For  i← k,2*n+1 execute
                 Write  a[i-n+1][n+k-i] + ',';
             EndFor
      EndIf
EndAlgorithm
```

**C.**

```
Algorithm c(n):                              {n=5 la apel}
    For  k← 1,2*n-1 execute
       If  k ≤  n
          then
             For  i← 1,k execute
                 Write a[n-k+1][i] + ','
             EndFor
          else
             For  i← k,2*n-1 execute
                 Write  a[i-k+1][n-i+1] + ',';
             EndFor
      EndIf
EndAlgorithm
```

**D.**

```
Algorithm d(n):                                {n=5 la apel}
    For  k← 1,2*n-1 execute
       If  k ≤  n
          then
             For  i← 1,k execute
                 Write a[i][k+1-i] + ','
             EndFor
          else
             For  i← k,2*n-1 execute
                 Write  a[i-n+1][n+k-i] + ',';
             EndFor
      EndIf
EndAlgorithm
```

**3.11.** Avem matricea:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 |

Care din metodele următoare afişează elementele matricii în ordinea:

**1,7,13,19,25,31, 32,26,20,14,8,2, 3,9,15,21,27,33, 34,28,22,16,10,4 5,11,17,23,29,35, 36,30,24,18,12,6**

**A.**

```
Algorithm afisMatrice (n)                      {n=6 la apel}
       For i← 1,n execute
           If (i MOD 2) =1
              then
                 For  j← 1,n execute
                       Write [j][i] + ’,’
                 EndFor
              else
                 For(int j← n,1,-1  execute
                       Write a[j][i] + ’,’
                 EndFor
          EndIf
       EndFor
 EndAlgorithm
```

**B.**

```
Algorithm afisMatrice (n)                          {n=6 la apel}
       For i← 1,n execute
           If (i MOD 2) = 0
              then
                 For  j← 1,n execute
                       Write [i][n-j] + ',' 
                 EndFor
              else
                 For(int j← 1,n  execute
                       Write a[i][j] + ',' 
                 EndFor
          EndIf
       EndFor
 EndAlgorithm
```

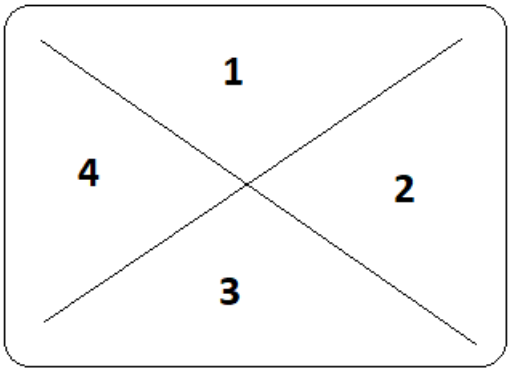
C.

```
Algorithm afisMatrice (n)                          {n=6 la apel}
       For i← 1,n execute
           If (i MOD 2) = 0
              then
                 For  j← 1,n execute
                       Write [n-j+1][i] + ',' 
                 EndFor
              else
                 For(int j← 1,n  execute
                       Write a[j][i] + ',' 
                 EndFor
          EndIf
       EndFor
 EndAlgorithm
```

D.

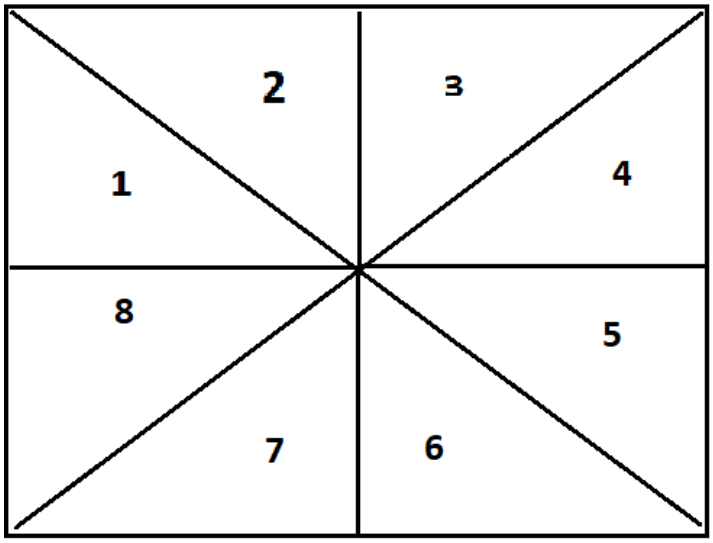
```
Algorithm afisMatrice (n)                          {n=6 la apel}
       For i← 1,n execute
           If (i MOD 2) = 1
              then
                 For  j← n,1,-1 execute
                       Write [n-j+1][i] + ',' 
                 EndFor
              else
                 For(int j← 1,n  execute
                       Write [n-j+1][i] + ',' 
                 EndFor
          EndIf
       EndFor
 EndAlgorithm
```

**3.12.** Fie o matrice pătratică cu zonele **1**,**2**,**3**,**4** ca în figura ce urmează. Fiecare zona nu conţine elementele matricii aflate pe cele două diagonale. Care afirmaţii sunt adevărate?




**A.** pentru **n=5** zona **1** contine **5** elemente;
**B.** pentru **n=100** zona **2** conţine **2450** elemente;
**C.** pentru **n=99** zona **3** conţine $49^2$ elemente;
**D.** pentru **n>1,** cele **4** zone conţin în total **n*(n-2)-(n%2)** elemente.

**3.13.** Fie o matrice pătratică cu zonele **1,2,3,4,5,6,7,8** ca în figura următoare. Fiecare zonă nu conţine elementele matricii aflate pe cele două diagonale şi pe cele 2 axe de simetrie.
Atenţie: dacă **n** este par cele două axe de simetrie nu conţin nici un element, dacă **n** este impar atunci axele de simetrie conţin elemente ale matricii. Care afirmaţii sunt adevărate?




**A.** Elementele din zona **1** sunt în număr de **($n^2$ – 4*n) DIV 8**, pentru **n>1**;
**B.** Elementele din zona **2** sunt în număr de **3** elemente, pentru **n=7**;
**C.** Elementele din zona **3** sunt în număr de **($n^2$-2*n*(1+(n MOD 2)) + 2*(n MOD 2)) DIV 8**, **n>1**;
**D.** Elementele din zona **4** sunt în număr de **($n^2$-2*n*(1+(n MOD 2)) + 3*(n MOD 2)) DIV 8, n>1**.

**3.14.** Ce face algoritmul citR(...)?

```
                          {m,n>1 naturale si m,n <10, n2=n, la apel primar}
                          {a este o marice de m linii şi n cooane}
Algorithm citR(m,n,n2,a):
  If m > 0
      then
         If  n > 0
            then
                  citR(m,n-1,n2,a)
                  Write "a[" + m + "][" + n +"]="
                  Citeşte  a[m][n]
            else
                  citR(m-1,n2,n2,a)
         EndIf
  EndIf
EndAlgorithm
```

**A.** Citeşte recursiv o matrice coloană după coloană începând cu prima coloană;
**B.** Citeşte recursiv o matrice linie dupa linie începând cu prima linie;
**C.** Citeşte recursiv o matrice linie dupa linie începând cu ultima linie;
**D.** Citeşte recursiv o matrice coloană după coloană începând cu ultima coloană.

**3.15.** Ce elemente are matricea pătratică **a** de ordinul **n**(linie cu linie) dupa execuţia următorului algoritm:

```
Algorithm s(n):
    k← 0
    For i← 1,(n DIV 2)+1 execute
           For j← i,n-i+1 execute
                 k← k+1
                 a[i][j]← k
           EndFor
           For j← i+1,n-i+1 execute
                 k← k+1
                 a[j][n-i+1]← k
           EndFor
           For j←n-i,i,-1 execute
                 k← k+1
                 a[n-i+1][j]← k
           EndFor
           For j← n-i,i+1,-1 execute
                 k← k+1
                 a[j][i]← k
           EndFor
    EndFor
EndAlgorithm
```

**A. n =5 ⇒ 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25;**
**B. n =5 ⇒ 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25;**
**C. n =5 ⇒ 1,6,11,16,21, 2,7,12,17,22, 3,8,13,18,23, 4,9,14,19,24, 5,10,15,20,25;**
**D. n =5 ⇒ 1,2,3,4,5, 16,17,18,19,6, 15,24,25,20,7, 14,23,22,21,8, 13,12,11,10,9.**

**3.16.** O matrice cu 8 linii, formată doar din elemente 0 și 1, are următoarele trei proprietăți: (model FMI)

a. prima linie conține un singur element cu valoarea 1,
b. linia j conține de două ori mai multe elemente nenule decât linia j − 1, pentru orice $j \in \{2, 3, \ldots, 8\}$,
c. ultima linie conține un singur element cu valoarea 0.

Care este numărul total de elemente cu valoarea 0 din matrice?

**A. 528;**
**B. 769;**
**C. 777;**
**D. nu există o astfel de matrice.**

**3.17.** Se dorește afișarea unui pătrat împreună cu interiorul său folosind doar caracterele * (asterisc) și . (punct) (conform desenului de mai jos). Exemplul din desen ilustrează un pătrat având laturile de ***n*** = 6 asteriscuri şi apoi pătratele interioare de puncte şi asteriscuri. (Un pătrat de astericuri apoi un pătrat de puncte, etc). Pentru acesta a fost necesară utilizarea a 24 asteriscuri și 12 puncte.

```
* * * * * *
* . . . . *
* . * * . *
* . * * . *
* . . . . *
* * * * * *
```

Care din afirmațiile de mai jos sunt adevărate?

**A.** Pentru **n = 7,** este nevoie de exact **36** asteriscuri și **13** puncte**.**
**B**. Pentru **n = 7**, este nevoie de exact **32** asteriscuri și **17** puncte.
**C**. Pentru **n = 8**, este nevoie de exact **40** asteriscuri și **24** puncte.
**D**. Pentru **n =18**, este nevoie de exact **200** asteriscuri și **124** puncte.