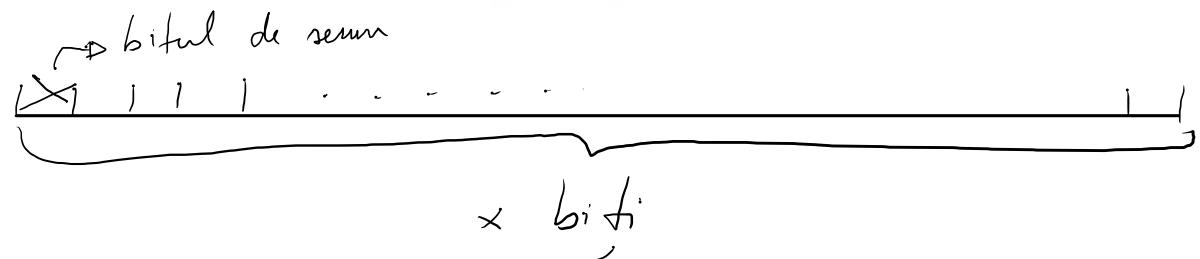


→ vom reține valori cu SEM N

4. Un tip de date întreg reprezentat pe x biți (x este număr natural strict pozitiv) va putea reține valori întregi din:

- A. $[0, 2^x]$
- B. $[0, 2^{x-1}-1]$
- C. $[-2^{x-1}, 2^{x-1}-1]$
- D. $[-2^x, 2^x-1]$



⇒ vom avea $x-1$ biți liberi pentru stocarea efectivă a valoii

↳ cea mai mare valoare este: $2^{x-1} - 1$

↳ cea mai mică valoare este: $-2^{x-1} + 1 - 1 = -2^{x-1}$

⇒ tipul nostru de date stocăresc din: $[-2^{x-1}, 2^{x-1}-1]$

Ref: Cursul 4

6. Se consideră următoarea expresie logică: $(\text{NOT } Y \text{ OR } Z) \text{ OR } (X \text{ AND } Y)$. Alegeți valorile pentru X, Y, Z astfel încât rezultatul evaluării expresiei să fie *adevărat*:

	$\neg Y \text{ OR } Z$	$X \text{ AND } Y$	$E(x)$
A. $X \leftarrow \text{fals}; Y \leftarrow \text{fals}; Z \leftarrow \text{fals};$	T	F	T
B. $X \leftarrow \text{fals}; Y \leftarrow \text{adevărat}; Z \leftarrow \text{fals};$	F	F	F
C. $X \leftarrow \text{adevărat}; Y \leftarrow \text{fals}; Z \leftarrow \text{adevărat};$	T	F	T
D. $X \leftarrow \text{fals}; Y \leftarrow \text{adevărat}; Z \leftarrow \text{adevărat};$	T	F	T

\Rightarrow A, C, D

m.	x	y	z	$\neg y \text{ OR } z$	$x \text{ AND } y$	$E(x)$
1	T	T	T	.	.	-
2	T	T	F	.	.	-
3	T	F	T	.	.	-
4	T	F	F	.	.	-
5	F	T	T	.	.	-
6	F	T	F	.	.	-
7	F	F	T	.	.	-
8	F	F	F	.	.	-

7. 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: $\hookrightarrow \underline{n \%_3 = 0 \text{ și } (n \%_{10} = 4 \text{ sau } n \%_{10} = 6)}$

~~A.~~ $n \%_3 = 0$ și $(n \%_{10} = 4 \text{ sau } n \%_{10} = 6)$

B. $n \%_3 = 0$ și $(n \%_{10} = 4 \text{ sau } n \%_{10} = 6)$

C. $(n \%_3 = 0 \text{ și } n \%_{10} = 4)$ sau $(n \%_3 = 0 \text{ și } n \%_{10} = 6)$

~~D.~~ $(n \%_3 = 0 \text{ și } n \%_{10} = 4)$ sau $n \%_{10} = 6$

B, C

5. Se dă subalgoritmul $f(a, b)$:

→ Ce face?

```
Subalgoritm f(a, b):
    Dacă a > 1 atunci
        returnează b * f(a - 1, b)
    altfel
        returnează b * f(a + 1, b)
    SfDacă
SfSubalgoritm
```

} se vențe

→ 4U

de cod este o funcție recursive
→ un loop
+ condiție de oprire
oare conditie de oprire \Rightarrow Bucle / Ciclu Infinit

Precizați de câte ori se auto apelează subalgoritmul $f(a, b)$ în urma execuției următoarei secvențe de instrucțiuni:

```
a ← 4, b ← 3
c ← f(a, b)
```

- A. de 4 ori
- B. de 3 ori
- C. de o infinitate de ori
- D. niciodată

C \rightarrow Bucle infinit

1. Subalgoritmul generare(n) prelucrează un număr natural n ($0 < n < 100$).

```
Subalgoritm generare(n):
    nr ← 0
    Pentru i ← 1, 1801 execută
        folositi ← fals
    SfPentru
    CâtTimp nu folositn execută
        suma ← 0, folositn ← adevărat
        CâtTimp (n ≠ 0) execută
            cifra ← n MOD 10, n ← n DIV 10
            suma ← suma + cifra * cifra * cifra
        SfCâtTimp
        n ← suma, nr ← nr + 1
    SfCâtTimp
    returnează nr
SfSubalgoritm
```

Precizați care este efectul acestui subalgoritmul. → Ce face?

- Calculează, în mod repetat, suma cuburilor cifrelor numărului n până când suma egalează numărul n și returnează numărul repetărilor efectuate
- Calculează suma cuburilor cifrelor numărului n și returnează această sumă
- 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 acest număr

} → suma cuburilor cifrelor lui n

→ îl înlocuiește pe n cu suma cuburilor cifrelor sale pînă când se obține o valoare de calculat anterior sau la n - un Δ .

Returnează numărul de înlocuire.

3. Se consideră subalgoritmul expresie(n), unde n este un număr natural ($1 \leq n \leq 10000$).

Subalgoritm expresie(n):

Dacă $n > 0$ atunci

Dacă $n \bmod 2 = 0$ atunci

returnează $-n * (n + 1) + \underline{\text{expresie}(n - 1)}$

altfel

returnează $n * (n + 1) + \underline{\text{expresie}(n - 1)}$

SfDacă

altfel

returnează 0 $\rightarrow u = 0 \Rightarrow 0$

SfDacă

SfSubalgoritm

$$\left\{ \begin{array}{l} \rightarrow 1 * 2 - 2 * 3 + 3 * 4 - 4 * 5 + \dots \stackrel{+/-}{=} n * (n+1) \\ \left\{ \begin{array}{l} 1 * 2 - 2 * 3 + \dots + (-1)^{n+1} * n * (n+1) \\ 1 * 2 - 2 * 3 + \dots - (-1)^n * n * (n+1) \end{array} \right. \end{array} \right.$$

Precizați forma matematică a expresiei $E(n)$ calculată de acest subalgoritm:

A. $E(n) = 1 * 2 - 2 * 3 + 3 * 4 + \dots + \underline{(-1)^{n+1}} * n * (n + 1)$

B. $E(n) = 1 * 2 - 2 * 3 + 3 * 4 + \dots + \underline{(-1)^n} * n * (n + 1)$

C. $E(n) = \underline{1 * 2 + 2 * 3 + 3 * 4 + \dots + (-1)^{n+1}} * n * (n + 1)$

D. $E(n) = \underline{1 * 2 - 2 * 3 - 3 * 4 - \dots - (-1)^n} * n * (n + 1)$

A

8. Fie următorul subalgoritm:

Subalgoritm f(a):

Dacă $a \neq 0$ atunci

returnează $a + f(a - 1)$

altfel

returnează 0 $\rightarrow a = 0$

SfDacă

SfSubalgoritm

$$1 + 2 + \dots + a = \frac{a * (a + 1)}{2}$$

Care din afirmațiile de mai jos sunt false?

← Ce făc?

- A daca a este negativ, subalgoritmul întoarce 0
- B valoarea calculată de f este $a * (a + 1) / 4$
- C subalgoritmul calculează suma numerelor naturale mai mici sau egale cu a
- ~~D~~ apelul $f(-5)$ intră în ciclu infinit.

A, B, C

9. Se consideră următorul subalgoritm:

```
Subalgoritm SA9(a):
    Dacă a < 50 atunci
        Dacă a MOD 3 = 0 atunci
            returnează SA9(2 * a - 3)
        altfel
            returnează SA9(2 * a - 1)
    SfDacă
    altfel
        returnează a → C. d. oprire
    SfDacă
SfSubalgoritm
```

$$f(a) = \begin{cases} f(2 \cdot a - 3), & a \% 3 = 0 \\ f(2 \cdot a - 1), & a \% 3 \neq 0 \\ a, & a \geq 50 \end{cases}$$

Formule matematice

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

- A 16
- B 61
- C 4
- D 31

A, B, D

← Ce face ?

$$f(16) \rightarrow \underline{f(31)} \rightarrow f(61) \rightarrow 61$$

$$f(4) \rightarrow f(7) \rightarrow f(13) \rightarrow f(25) \rightarrow f(53) \rightarrow f(97) \rightarrow 97$$

10. Se consideră subalgoritmul prelucreaza(v , k), unde v este un sir cu k numere naturale ($1 \leq k \leq 1000$).

```

Subalgoritm prelucreaza(v, k)
    i ← 1, n ← 0
    CâtTimp i ≤ k și vi ≠ 0 execută
        y ← vi, c ← 0
        CâtTimp y > 0 execută
            Dacă y MOD 10 > c atunci
                c ← y MOD 10
            SfDacă
            y ← y DIV 10
        SfCâtTimp
        n ← n * 10 + c
        i ← i + 1
    SfCâtTimp
    returnează n
SfSubalgoritm

```

} cifre maxime a lui y

↳ construiesc numărul

} Construiesc un număr care conține
cifre maxime a fiecărui număr din
sir până la numărul o săn finalul
sirului.

Precizați pentru care valori ale lui v și k subalgoritmul returnează valoarea 928.

- A $v = (194, 121, 782, 0)$ și $k = 4$
- B $v = (928)$ și $k = 1$
- C $v = (9, 2, 8, 0)$ și $k = 4$
- D $v = (8, 2, 9)$ și $k = 3$

↳ Ce face ?

A, C

11. Se consideră următoarea expresie logică $(X \text{ OR } Z) \text{ AND } (\text{NOT } X \text{ OR } Y)$. Alegeti valorile pentru X , Y , Z astfel încât evaluarea expresiei să dea rezultatul TRUE:

	x	$\neg z$	$\neg x$	y	$E(x)$
A. $X \leftarrow \text{FALSE}; Y \leftarrow \text{FALSE}; Z \leftarrow \text{TRUE};$	T	F	F	F	F
B. $X \leftarrow \text{TRUE}; Y \leftarrow \text{FALSE}; Z \leftarrow \text{FALSE};$	F	T	T	F	F
C. $X \leftarrow \text{FALSE}; Y \leftarrow \text{TRUE}; Z \leftarrow \text{FALSE};$	F	T	T	T	F
D. $X \leftarrow \text{TRUE}; Y \leftarrow \text{TRUE}; Z \leftarrow \text{TRUE};$	T	F	F	T	T

A, D

12. Se consideră următorul program:

Varianta C	Varianta C++	Varianta Pascal
<pre>#include <stdio.h> int prelVector(int v[], int *n) { int s = 0; int i = 2; while (i <= *n) { s = s + v[i] - v[i - 1]; if (v[i] == v[i - 1]) *n = *n - 1; i++; } return s; } int main(){ int v[8]; v[1] = 1; v[2] = 4; v[3] = 2; v[4] = 3; v[5] = 3; v[6] = 10; v[7] = 12; int n = 7; int rezultat = prelVector(v, &n); printf("%d;%d", n, rezultat); return 0; }</pre>	<pre>#include <iostream> using namespace std; int prelVector(int v[], int&n) { int s = 0; int i = 2; while (i <= n) { s = s + v[i] - v[i - 1]; if (v[i] == v[i - 1]) n -= 1; i++; } return s; } int main(){ int v[8]; v[1] = 1; v[2] = 4; v[3] = 2; v[4] = 3; v[5] = 3; v[6] = 10; v[7] = 12; int n = 7; int rezultat = prelVector(v, n); cout << n << ":" << rezultat; return 0; }</pre>	<pre>type vector=array [1..10] of integer; function prelVector(v: vector; var n: integer): integer; var s, i: integer; begin s := 0; i := 2; while (i <= n) do begin s := s + v[i] - v[i - 1]; if (v[i] = v[i - 1]) then n := n - 1; i := i + 1; end; prelVector := s; end; var n, rezultat:integer; v:vector; begin n := 7; v[1] := 1; v[2] := 4; v[3] := 2; v[4] := 3; v[5] := 3; v[6] := 10; v[7] := 12; rezultat := prelVector(v, n); write(n, ':', rezultat); end.</pre>

Precizați care este rezultatul afișat în urma executării programului.

X 7;11
B 6;9
X 7;9
X 7;12

B

↳ 6 face ?

Se calculează suma diferențelor dintre oricare 2 termeni consecutivi în sir, dacă doar jumătate 2 termeni consecutivi egali, se obțin și rezultatul

$$V\{ \} = \{ 1, 3, 2, \underline{3}, \underline{3}, 10, \cancel{12} \}$$

$$n = 7 - 1 = \underline{6}$$

$$\text{sum} = 3 + (-2) + 1 + 0 + 7 = \underline{9}$$

$$R: 6; 9$$

15. Se consideră toate șirurile de lungime $l \in \{1, 2, 3\}$ formate din litere din mulțimea $\{a, b, c, d, e\}$. Câte dintre aceste șiruri au elementele ordonate strict descrescător și un număr impar de vocale? (a și e sunt vocale)

- A. 14
- B. 7
- C. 81
- D. 78

A

↳ observăm că toate generările vor conține cel mult 1 vocală

$l=1$		$l=2$		$l=3$	
a		ba	eb	dc a	edc
	c	ca	ec	db a	ed b
e		da	ed	cb a	ec b
"	2	"	6	"	6

$l \in \{1, 2, 3\} \Rightarrow 14$ generații

16. Se consideră dat subalgoritmul $\text{apartine}(x, a, n)$ care verifică dacă un număr natural x aparține mulțimii a cu n elemente; a este un sir cu n elemente și reprezintă o mulțime de numere naturale ($1 \leq n \leq 200, 1 \leq x \leq 1000$).

Fie subalgoritmii $\text{reuniune}(a, n, b, m, c, p)$ și $\text{calcul}(a, n, b, m, c, p)$, descriși mai jos, unde a, b și c sunt siruri care reprezintă mulțimi de numere naturale cu n, m și respectiv p elemente ($1 \leq n \leq 200, 1 \leq m \leq 200, 1 \leq p \leq 400$). Parametrii de intrare sunt a, n, b, m și p , iar parametrii de ieșire sunt c și p .

<pre> 1. Subalgoritm reuniune(a, n, b, m, c, p): 2. Dacă <u>n = 0</u> atunci 3. Pentru i ← 1, m execută 4. p ← p + 1, <u>c_p ← b_i</u> (1) 5. SFpentru 6. altfel 7. Dacă nu <u>apartine(a_n, b, m)</u> atunci 8. p ← p + 1, <u>c_p ← a_n</u> (2) 9. SFdacă 10. → <u>reuniune(a, n - 1, b, m, c, p)</u> 11. SFdacă 12. SfSubalgoritm </pre>	<pre> 1. Subalgoritm calcul(a, n, b, m, c, p): 2. p ← 0 3. reuniune(a, n, b, m, c, p) — 4. SfSubalgoritm </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------

Precizați care dintre afirmațiile de mai jos sunt întotdeauna adevărate:

- Ce face?
- A. când mulțimea a conține un singur element, apelul subalgoritmului $\text{calcul}(a, n, b, m, c, p)$ provoacă apariția unui ciclu infinit
 - B. când mulțimea a conține 4 elemente, apelul subalgoritmului $\text{calcul}(a, n, b, m, c, p)$ provoacă executarea instrucțiunii de pe linia 10 a subalgoritmului reuniune de 4 ori
 - C. când mulțimea a conține 5 elemente, apelul subalgoritmului $\text{calcul}(a, n, b, m, c, p)$ provoacă executarea instrucțiunii de pe linia 2 a subalgoritmului reuniune de 5 ori
 - D. când mulțimea a are aceleași elemente ca și mulțimea b , în urma execuției subalgoritmului $\text{calcul}(a, n, b, m, c, p)$ mulțimea c va avea același număr de elemente ca și mulțimea a

B, D

(1) - Conținutul lo sirul e foarte elementele
din sirul b

(2) - Doar elementul care e din sirul
e un opere și în sirul b, conținutul
e c, elementul din e.

(3) - Conținutul e foarte elementele
distincțe din e și b

20. Fie subalgoritmul `factoriPrimi(n, d, k, x)` care determină cei k factori primi ai unui număr natural n , începând căutarea factorilor primi de la valoarea d . Parametrii de intrare sunt numerele naturale n , d și k , iar parametrii de ieșire sunt sirul x cu cei k factori primi ($1 \leq n \leq 10000$, $2 \leq d \leq 10000$, $0 \leq k \leq 10000$).

```
Subalgoritm factoriPrimi(n, d, k, x):
    Dacă n MOD d = 0 atunci
        k ← k + 1
        x[k] ← d
    SfDacă
    Cât timp n MOD d ≠ 0 execută
        n ← n DIV d
    SF Cât Timp
    Dacă n > 1 atunci
        factoriPrimi(n, d + 1, k, x)
    SF Dacă
Sf Subalgoritm
```

Stabilii de câte ori se autoapelă subalgoritmul `factoriPrimi(n, d, k, x)` în următoarea secvență de instrucțiuni:

```
n ← 120
d ← 2
k ← 0
factoriPrimi(n, d, k, x)
```

$\text{factoriPrimi}(120, 2, 0, \{\}) \rightarrow \text{factoriPrimi}(15, 3, 1, \{2\})$

- A) de 3 ori
 B) de 5 ori
 C) de 6 ori

D) de același număr de ori ca și în cadrul secvenței de instrucțiuni:

```
n ← 750
d ← 2
k ← 0
factoriPrimi(n, d, k, x)
```

$\text{factoriPrimi}(5, 3, 2, \{2, 3\}) \rightarrow \text{factoriPrimi}(5, 5, 2, \{2, 3\})$

Opiniu : $M = 1$
 $d = 5$

$k = 3$

$x = [2, 3, 5]$

$$\begin{array}{c|c}
750 & 2 \\
375 & 3 \\
125 & 5 \\
25 & 5 \\
5 & 5 \\
1 & \rightarrow \text{stop}
\end{array} \quad 750 = 2 \cdot 3 \cdot 5^3$$

23. Fie s un sir de numere naturale unde elementele s_i sunt de forma $s_i = \begin{cases} x, & \text{dacă } i = 1 \\ x + 1, & \text{dacă } i = 2 \\ s_{(i-1)} @ s_{(i-2)}, & \text{dacă } i > 2 \end{cases}$

($i = 1, 2, \dots$). Operatorul $@$ concatenează cifrele operandului stâng cu cifrele operandului drept, în această ordine (cifre aferente reprezentării în baza 10), iar x este un număr natural ($1 \leq x \leq 99$). De exemplu, dacă $x = 3$, sirul s va conține valorile $3, 4, 43, 434, 43443, \dots$. Precizați numărul cifrelor aceluia termen din sirul s care precede termenul format din k ($1 \leq k \leq 30$) cifre.

dacă $x = 15$ și $k = 6$, numărul cifrelor termenului aflat în sirul s în fața termenului format din k cifre este 5.

B dacă $x = 2$ și $k = 8$, numărul cifrelor termenului aflat în sirul s în fața termenului format din k cifre este 5.

C dacă $x = 14$ și $k = 26$, numărul cifrelor termenului aflat în sirul s în fața termenului format din k cifre este 16.

dacă $x = 5$ și $k = 13$, numărul cifrelor termenului aflat în sirul s în fața termenului format din k cifre este 10.

15, 16, 1615, ...
2, 2, 4, 6, 10, 16
→ 26

B, C

3, 4, 43, 434, 43443, 4344343, ...

1, 1, 2, 3, 5, 8, 13, ...

$\ell_{(1)} = \ell_{(1-1)} + \ell_{(1-2)} \rightarrow$ sirul lui Fibonacci

26. Se consideră subalgoritmul $\text{alg}(x, b)$ cu parametrii de intrare două numere naturale x și b ($1 \leq x \leq 1000, 1 < b \leq 10$).

```

Subalgoritm alg(x, b):
    s ← 0
    CâtTimp x > 0 execută
        s ← s + x MOD b
        x ← x DIV b
    SfCâtTimp
    returnează s MOD (b - 1) = 0
SfSubalgoritm
  
```

} → sumă cifrelor lui x în
base b

→ T - dacă sumă cifrelor lui x în base b se divide cu $(b - 1)$
F - în caz contrar

Precizați efectul acestui subalgoritm. → Ce face?

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

* Acosta problemă ilustrează Generalizarea criteriului de divizibilitate cu scris în baza 10 către orice altă bază de numerație unde dacă

$$\text{sum cif } (x_{(b)}) \% (b - 1) = 0 \Rightarrow x \% (b - 1) = 0$$

28. Dreptunghiul cu laturile de lungimi m și n (m, n – numere naturale, $0 < m < 101$, $0 < n < 101$) este împărțit în pătrățele cu latura de lungime 1. Se consideră subalgoritmul dreptunghi(m, n):

```

Subalgoritm dreptunghi(m, n)
    d ← m
    c ← n
    CâtTimp d ≠ c execută
        Dacă d > c atunci
            d ← d - c
        altfel
            c ← c - d
        SfDacă
    SfCâtTimp
    returnează m + n - d
SfSubalgoritm
  
```

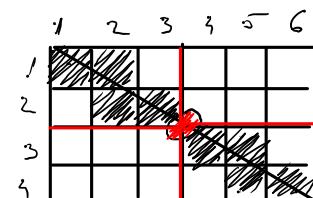
$\left. \begin{array}{l} \\ \\ \\ \\ \\ \end{array} \right\}$ cel mai mare divizor comun $\rightarrow d = c = \text{cmmdc}(m, n)$
 $\rightarrow m + n - \text{cmmdc}(m, n)$

Precizați efectul acestui subalgoritmul.

\hookrightarrow Ce face?

- A. Calculează și returnează numărul pătrățelor cu latura de lungime 1 traversate de o diagonală a dreptunghiului.
- X Determină în d cel mai mare divizor comun al laturilor dreptunghiului și returnează diferența dintre suma laturilor dreptunghiului și d .
- C Dacă $m = 8$ și $n = 12$, returnează 16.
- X Dacă $m = 6$ și $n = 11$, returnează 15.

A, C



$$10 - 2 = 8$$



$$4 + 5 = 9 - 1 = 8$$

30. O matrice cu 8 linii, formată doar din elemente 0 și 1, are următoarele trei proprietăți:

- 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 $j \in \{2, 3, \dots, 8\}$,
- c. ultima linie conține un singur element cu valoarea 0.

Care este numărul total de elemente cu valoarea 0 din matrice?

A 777

B 769

C 528

D nu există o astfel de matrice

A

$$l_1 : 1 - \text{un} \text{u}$$

$$l_2 : 2 - \text{un} \text{u}$$

...

$$l_8 : 2^7 - \text{un} \text{u}, \quad 1 - \text{zero} \quad (1)$$

$$\begin{aligned} \text{Dim (1)} \Rightarrow \text{nr. coloane} &= 2^7 + 1 = 128 + 1 = 129 \\ \text{nr. linii} &= 8 \end{aligned} \quad \left. \begin{array}{l} \text{ } \\ \Rightarrow \text{numărul de elemente} \end{array} \right\} = 8 \cdot 129 = 1032$$

$$\text{Câte elemente 1 sunt?} : 1 + 2 + 4 + \dots + 2^7 = 1 \cdot \frac{2^8 - 1}{2 - 1} = 2^8 - 1 = 255$$

$$\Rightarrow \text{Valoare "0"} = 1032 - 255 = 777$$