

# Explicații pentru laboratorul #1

Indicatii date de profesoara:

1. Valorile pentru datele de intrare se vor introduce cate una pe linie
2. Pentru rezolvarea problemelor nu sunt necesare structuri repetitive sau conditionale (deci nu le vom folosi in rezolvare).

## Problema #1 laboratorul #1

Se citesc de la tastatură  $n$ ,  $p_1$ ,  $p_2$ ,  $p_3$  numere naturale, cu  $p_1$ ,  $p_2$ ,  $p_3$  prime. Să se afișeze câte numere naturale mai mici sau egale cu  $n$  sunt divizibile cu  $p_1$ ,  $p_2$ , sau  $p_3$ .

**Exemplu:** pentru  $n = 2025$ ,  $p_1 = 2$ ,  $p_2 = 5$ ,  $p_3 = 7$  se va afișa un mesaj de forma: Numărul valorilor mai mici sau egale cu 2025 care se divid cu 2, 5 sau 7 este 1331

## Rezolvarea problemei #1 laboratorul #1

Trebuie sa rezolvam aceasta problema fara structuri repetitive (while, for) si fara conditionale (if).

Asta ne spune ca exista o **formula matematica** pentru a determina rezultatul cerut. Vom folosi un principiu matematic din combinatorica.

### Principiul includerii si excluderii

Consideram urmatoarele multimi:

$$A = \text{multimea numerelor } \leq n \text{ divizibile cu } p_1$$

$$B = \text{multimea numerelor } \leq n \text{ divizibile cu } p_2$$

$$C = \text{multimea numerelor } \leq n \text{ divizibile cu } p_3$$

Ne dorim sa aflam numarul de elemente din **reuniunea** acestor multimi, adica

$$|A \cup B \cup C|.$$

Formula matematica pentru aflarea cardinalului reuniunii de mai sus este:

$$|A \cup B \cup C| = |A| + |B| + |C| - (|A \cap B| + |A \cap C| + |B \cap C|) + |A \cap B \cap C|$$

**Explicatia formulei:**

1. Adunam cardinalele celor trei multimi:

$$|A| + |B| + |C|$$

- Problema!: Elementele comune (intersectiile) au fost numarate de mai multe ori (de exemplu, cand un numar  $\leq n$  este divizibil si cu  $p_1$  si cu  $p_2$ )

2. Scadem intersectiile de cate doua multimi

$$-(|A \cap B| + |A \cap C| + |B \cap C|)$$

- Problema!: Daca un numar este divizibil si cu  $p_1$ , si cu  $p_2$ , si cu  $p_3$  (deci se afla in  $A \cap B \cap C$ ):
- La pasul 1, a fost adunat de 3 ori (o data pentru  $A$ , o data pentru  $B$  si inca o data pentru  $C$ )
- La pasul 2, a fost scazut de 3 ori (o data pentru  $A \cap B$ , o data pentru  $A \cap C$ , o data pentru  $B \cap C$ )
- Rezultatul net este 0. Acest numar nu este numarat deloc

3. Adunam la loc intersectia celor trei multimi

$$+ |A \cap B \cap C|$$

## Aplicarea formulei in problema

**Pasul #1: aflam cardinalele lui A, B si C (adica cate numere  $\leq n$  sunt divizibile cu  $p_1, p_2, p_3$ , pe rand)**

Cum aflam cate numere naturale mai mici sau egale cu  $n$  sunt divizibile cu un numar natural  $p$ ?

Reformulam problema astfel: cate numere naturale  $\leq n$  sunt multipli de  $p$ ?

Multiplii de  $p$  arata asa:  $1 * p, 2 * p, 3 * p, \dots, k * p, \dots$

Ce cautam noi? - Cel mai mare numar intreg  $k$  (care reprezinta al catelea multiplu este) pentru care  $k * p$  este inca  $\leq n$ :

Vrem cel mai mare  $k \in \mathbb{N}$  a. i.  $k * p \leq n$

Impartind in inegalitate cu  $p$ , rezulta

Vrem cel mai mare  $k \in \mathbb{N}$  a. i.  $k \leq \frac{n}{p}$

Dar acel  $k$  cautat este *fix partea intreaga a lui  $\frac{n}{p}$* .

Operatorul `//` din Python calculeaza exact partea intreaga a impartirii.

```
div_p1 = n // p1
div_p2 = n // p2
div_p3 = n // p3
```

**Pasul #2: aflam cardinalele multimilor  $A \cap B, A \cap C, B \cap C$**

Un numar natural este divizibil cu doua numere prime daca este divizibil cu produsul lor:

```
div_p1_p2 = n // (p1 * p2)
div_p1_p3 = n // (p1 * p3)
div_p2_p3 = n // (p2 * p3)
```

**Pasul #3: aflam cardinalul multimii  $A \cap B \cap C$**

```
div_p1_p2_p3 = n // (p1 * p2 * p3)
```

**Pasul #4: aplicam formula**

```
rezultat = div_p1 + div_p2 + div_p3 - (div_p1_p2 + div_p1_p3 + div_p2_p3) +
div_p1_p2_p3
```

```
# Solutia problemei #1 din laboratorul #1
n = int(input("n: "))
p1 = int(input("p1: "))
p2 = int(input("p2: "))
p3 = int(input("p3: "))
```

```

div_p1 = n // p1      # Cardinalul multimii A (cate numere se divid cu p1)
div_p2 = n // p2      # Cardinalul multimii B (cate numere se divid cu p2)
div_p3 = n // p3      # Cardinalul multimii C (cate numere se divid cu p3)

div_p1_p2 = n // (p1 * p2)      # Cardinalul multimii A intersectat cu B
div_p1_p3 = n // (p1 * p3)      # Cardinalul multimii A intersectat cu C
div_p2_p3 = n // (p2 * p3)      # Cardinalul multimii B intersectat cu C

div_p1_p2_p3 = n // (p1 * p2 * p3)      # Cardinalul multimii A intersectat cu B intersectat cu C

rezultat = div_p1 + div_p2 + div_p3 -  

    (div_p1_p2 + div_p1_p3 + div_p2_p3) + div_p1_p2_p3
print(f"Numarul valorilor mai mici sau egale cu {n} care sunt divizibile cu {p1}, {p2}  

    sau {p3} este {rezultat}.")

```

## Problema #2 laboratorul #1

Se citește  $n \leq 1000$  natural. Să se afișeze numărul de zerouri aflate la finalul lui  $n!$ , fără a calcula factorialul.

**Exemplu:** Numărul de zerouri de la finalul lui  $987!$  este 244

Amintim ca obiectivul nostru este să nu folosim loop-uri (for, while) și if.

## Rezolvarea problemei #2 laboratorul #1

**Idee:** Un zero la final apare când  $n!$  are un factor egal cu 10. Dar  $10 = 2 * 5$ . În  $n!$ , factorii de 2 sunt mai mulți decât cei de 5, deci numărul de zerouri = cati factori de 5 apar în descompunerea lui  $n!$ .

Asadar, am redus problema la a găsi numărul de factori de 5 din descompunerea lui  $n! = 1 * 2 * 3 * \dots * n$ .

**Exemplu**  $n = 12 \implies n! = 1 * 2 * 3 * 4 * \underline{5} * 6 * 7 * 8 * 9 * \underline{10} * 11 * 12 = 479001600$  (avem 2 multipli de 5 deci 2 zerouri la final)

**Problema!** Ce se întâmplă pentru  $26!$ ?

$$n = 26 \implies n! = 1 * 2 * 3 * 4 * \underline{5} * 6 * 7 * 8 * 9 * \underline{10} * 11 * 12 * 13 * 14 * \underline{15} * 16 * 17 * 18 * 19 * \underline{20} * 21 * 22 * 23 * 24 * \underline{25} * 26$$

Avem 5 multipli de 5, deci 5 zerouri, nu?

```

import math
math.factorial(26)

```

Ups! sunt 6 zerouri, nu 5.

Problema este că pentru numărul 25 din componenta lui  $26!$ , am numărat doar un factor divizibil cu 5, când în realitate ar fi trebuit să îl număram de două ori deoarece  $25 = 5^2$ .

Aceeași problema apoi dacă ajungem la factori  $\geq 5^3 = 125$  etc. Va trebui să îl număram după puterea lui 5.

## Solutie

Numaram cati factori de 5 avem astfel:

- Multiplii de 5 aduc cate un 5  $\Rightarrow \left[ \frac{n}{5} \right]^{****}$
- Multiplii de 25 aduc cate doi de 5  $\Rightarrow \left[ \frac{n}{25} \right]$
- Multiplii de 125 aduc cate trei de 5  $\Rightarrow \left[ \frac{n}{125} \right]$
- Si asa mai departe...

Dar, din ipoteza problemei,  $n \leq 1000$ , deci puterea maxima a lui 5 va fi 4 ( $5^4 = 625 < 1000$ ,  $5^5 = 3125 > 1000$ , adica  $5^5$  depaseste constrangerea data).

\*\*\*\*: Am aratat ca numarul de multipli de  $p \leq n \in \mathbb{N}$  este  $\left[ \frac{n}{p} \right]$  (parte intreaga).

In final, am obtinut ca:

$$\forall n \in \mathbb{N} \text{ cu } n \leq 1000, \text{ zerouri}(n!) = \left[ \frac{n}{5} \right] + \left[ \frac{n}{25} \right] + \left[ \frac{n}{125} \right] + \left[ \frac{n}{625} \right]$$

Pentru  $n = 987$ :

- $\left[ \frac{987}{5} \right] = 197$
- $\left[ \frac{987}{25} \right] = 39$
- $\left[ \frac{987}{125} \right] = 7$
- $\left[ \frac{987}{625} \right] = 1$
- total: 244 zerouri

```
# Solutia problemei #2 laboratorul #1
n = int(input("n: "))
nr_zerouri = n // 5 + n // 25 + n // 125 + n // 625
print(nr_zerouri)
```

## Problema #3 laboratorul #1

Se citește un număr natural de la tastatură. Să se determine **înălțimea maximă a unei piramide ce se poate construi folosind maxim n caractere „\*”** astfel încât pe nivelul i al piramidei să se afle i \* . Să se afișeze și numărul de \* rămase.

**Exemplu:** pentru  $n = 12$  se va afișa mesajul Se poate construi o piramidă cu 4 niveluri având la dispoziție 12 caractere de tip \*. Vor rămâne neutilizate 2 caractere de tip \*. Explicație: Se folosesc 10 caractere de tip \* pentru construcția piramidei

```
*  
* *  
* * *  
* * * *
```

**Intuitie:**

Uitandu-ne la exemplul dat, vedem ca s-au folosit

$$1 + 2 + 3 + 4 = 10 \text{ stelute}$$

Asta e fix **suma Gauss!** - va fi notata  $S(h)$

Deci, ca sa alcatuim o piramida cu  $h$  etaje (h de la height) ne trebuie  $S(h) = \frac{h(h+1)}{2}$  stelute.

Asadar, noi va trebui sa determinam

$$h \in \mathbb{N} \text{ valoarea maxima pt. care } S(h) \leq n \quad (1)$$

Din nou, avem un "numar natural maxim  $\leq$  o valoare" - asta suna a parte intreaga (va aparea mai jos, la impartire).

Prelucram **inegalitatea (1)**:

$$\begin{aligned} S(h) \leq n &\iff \frac{h(h+1)}{2} \leq n \iff h(h+1) \leq 2n \\ h^2 + h - 2n &\leq 0 \quad (2) \end{aligned}$$

Rezolvam **inegalitatea (2)** cu necunoscuta  $h$  ca in liceu:

$$\begin{aligned} h^2 + h - 2n &= 0 \\ \Delta = 1 - 4 * (1 * (-2n)) &\iff \\ \Delta = 1 + 8n & \end{aligned}$$

Asadar,

$$h^2 + h - 2n = 0 \implies h_{1,2} = \frac{-1 \pm \sqrt{1 + 8n}}{2}$$

Dar

$$h \in \mathbb{N} \implies h \geq 0 \text{ si } h = \left[ \frac{-1 + \sqrt{1 + 8n}}{2} \right]$$

Am obtinut, deci, solutia pentru egalitatea  $h^2 + h - 2n = 0$ , de unde rezulta ca  $S(h) \leq n \iff \left[ \frac{-1 + \sqrt{1 + 8n}}{2} \right] \leq n$ .

In final,

$$h = \left[ \frac{-1 + \sqrt{1 + 8n}}{2} \right] \text{ valoarea maxima pt. care } S(h) \leq n, \forall n \in \mathbb{N}.$$

```
# Solutia problemei #3 laboratorul #1
import math
n = int(input("n: "))
h = (math.sqrt(1 + 8 * n) - 1) // 2
folosite = h * (h + 1) // 2
ramase = n - folosite
print(f"Se poate construi o piramida cu {h} niveluri, avand la dispozitie {n} caractere de tip *. Vor ramane neutilizate {ramase} caractere de tip *.")
```

## Problema #4 laboratorul #1

Să se verifice dacă un număr n citit de la tastatură este pătrat perfect (se va afișa True / False). Să se afișeze numărul de pătrate perfecte mai mici sau egale decât n.

## Rezolvarea problemei #4 laboratorul #1

$$n \text{ patrat perfect} \iff \exists k \in \mathbb{N} \text{ a. i. } n = k^2 \iff \exists k \in \mathbb{N} \text{ a. i. } k = \sqrt{n}$$

Deci,  $n \in \mathbb{N}$  e patrat perfect daca si numai daca  $\sqrt{n}$  este o valoare intreaga (naturala, mai exact).

Apoi, pentru cealalta cerinta, o reformulam in felul urmator: cautam **k maxim** a. i.  $k^2 \leq n$  (numerele cautate vor fi  $1^2, 2^2, 3^2, \dots, k_{max}^2$ ). Suna a ceva cunoscut?

Cautam cel mai mare numar natural  $k$  pentru care  $k^2 \leq n \iff k \leq \sqrt{n}$ . Dar partea intreaga ne da fix acel numar, deci

$$k_{max} = [\sqrt{n}]$$

Amintim ca notatia [ceva] inseamna "partea intreaga a lui ceva".

```
import math

n = int(input("n: "))
radacina_intreaga = math.isqrt(n)
e_patrat = (radacina_intreaga * radacina_intreaga == n)
print(f"n e patrat: {e_patrat}")

numar_patrate = radacina_intreaga
print(f"Numarul de patrate perfecte mai mici sau egale cu {n} este {numar_patrate}")
```

## Problema #5 laboratorul #1

Se citește un număr întreg de la tastatură. Să se determine dacă acesta este par sau impar folosind doar operatorii logici la nivel de biți (True / False).

## Rezolvarea problemei #5 laboratorul #1

In reprezentarea pe biti (adica in baza 2) a unui numar, ceea ce ne da paritatea lui este ultimul bit (se mai cheama si Least Significant Bit deoarece este corespunzator celei mai mici puteri ale lui  $2 \rightarrow 2^0$ ).

**Exemple:**

- 15 in baza 2 este 1111 (verificare:  $2^3 + 2^2 + 2^1 + 2^0 = 15$ )
- 6 in baza 2 este 110 (verificare:  $2^2 + 2^1 = 6$ )

Asadar, pentru rezolvarea problemei trebuie doar sa verificam daca ultimul bit (least significant bit) este 0 sau 1.

Exista o operatie care face fix acest lucru: operatia AND 1 (sau & 1 in Python).

De exemplu:

- 1111 & 1 = 1111 & 0001 = 0001 = 1
- 1010 & 1 = 1010 & 0001 = 0000 = 0

```
n = int(input("n: "))
```

```

if n & 1:      # adica if n & 1 == 1
    print(f"Numarul {n} este impar")
else:          # altfel ramane doar posibilitatea n & 1 == 0
    print(f"Numarul {n} este par")

```

## Problema #6 laboratorul #1

Să se verifice dacă un număr natural  $n$  este de forma  $2^k$  sau nu (True / False).

### Rezolvarea problemei #6 laboratorul #1

#### Metoda 1/3

Avem: Fie  $n \in \mathbb{N}$ . Atunci,

$$\exists k \in \mathbb{N} \text{ a.i. } n = 2^k \iff \log_2(n) \in \mathbb{N}$$

Deci,  $\log_2(n)$  trebuie să fie o valoare naturală.

```

# Solutia 1/3 a problemei #6 laboratorul #1
import math

n = int(input("n: "))

# Verificam ca n sa nu fie 0 deoarece, in cazul n = 0,
# codul care acum e in else ar da eroare
if n == 0:
    print(False)
else:
    if math.log2(n).is_integer():
        print(True)
    else:
        print(False)

```

#### Metoda 2/3

Considerăm reprezentarea numărului  $n$  în baza 2. Uite niste exemple de puteri ale lui 2 în baza 2:

- $1 \rightarrow 1$
- $2 \rightarrow 10$
- $4 \rightarrow 100$
- $8 \rightarrow 1000$
- $16 \rightarrow 1\ 0000$
- $256 \rightarrow 1\ 0000\ 0000$

Observăm că ce au în comun este că au exact un 1 în reprezentarea în baza 2.

Deci putem verifica dacă au exact un bit egal cu 1.

```

# Solutia 2/3 a problemei #6 laboratorul #1
n = int(input("n: "))
if bin(n).count("1") == 1:

```

```
print(True)
else:
    print(False)
```

## Metoda 3/3

Ne folosim de faptul ca daca  $n$  este o putere a lui 2, atunci  $n \& (n-1) = 0$ .

Stim ca daca  $n$  e putere a lui 2, atunci contine fix un bit egal cu 1. Uite ce se intampla cand scadem 1  $(n-1)$  si aplicam AND:

- $n = 16 \rightarrow 1\ 0000$ ,  $n-1 = 15 \rightarrow 0\ 1111$ ,  $1\ 0000 \& 0\ 1111 = 0\ 0000 = 0$
- $n = 64 \rightarrow 100\ 0000$ ,  $n-1 = 63 \rightarrow 011\ 1111$ ,  $100\ 0000 \& 011\ 1111 = 000\ 0000 = 0$

```
# Solutia 3/3 a problemei #6 laboratorul #1
n = int(input("n: "))
if n & (n-1) == 0:
    print(True)
else:
    print(False)
```