Programarea Algoritmilor
– SEMINAR NR. 3 –
(grupa 131)

1. Anagrame
   Se dau două șiruri formate din litere mici.
   Să se verifice dacă sunt sau nu anagrame.

   Exemplu: șirurile “emerit” și “treime” sunt anagrame,
   dar șirurile “emerit” și “treimi” nu sunt.

```
s = input("s = ")
t = input("t = ")
if len(s) != len(t):
    print("Nu sunt anagrame.")
else:
    pass # se inlocuieste "pass" cu una din solutiile de mai jos


    # Solutia (a): liste + sortare --> O(n * log(n))
    L1 = list(s)
    L2 = list(t)
    L1.sort() #O(n * log(n))
    L2.sort()
    if L1 == L2: #O(n)
        print("DA")
    else:
        print("NU")

    # Solutia (a')
    print("DA" if sorted(s)==sorted(t) else "NU")


    # Solutia (b): vectori de frecventa --> O(n)
    fs = [0]*26
    ft = [0]*26
    for i in s:
        fs[ord(i)-ord("a")] += 1
    for j in t:
        ft[ord(j)-ord("a")] += 1
    print("DA" if fs==ft else "NU")


    # Solutia (c): multimi + frecvente --> O(n*n)
    s1 = set(s)
    s2 = set(t)
    if s1 != s2:
        print("NU")
    else:
        for i in s1: #O(n) daca s are multe litere distincte
            if s.count(i) != t.count(i): #O(n)
                print("NU")
                break
        else:
            print("DA")

    # Solutia (d): dictionare --> O(n*n)
    # (O(n) lungime for * O(n) fct count) + O(n) creare dictionar
    d1 = {x:s.count(x) for x in set(s)}
    d2 = {y:t.count(y) for y in set(t)}
    print("DA" if d1==d2 else "NU")
```

```
# Solutia (e): dictionare pe post de vectori de frecventa --> O(n)
d1 = {}
for x in s:
    if x in d1:
        d1[x] += 1
    else:
        d1[x] = 1
d2 = {y:0 for y in set(t)}
for y in t:
    d2[y] += 1
print("DA" if d1==d2 else "NU")
```

TEMĂ: alte soluții pentru verificarea anagramelor:
(f) Cu un singur vector de frecventă ([0]*26), pentru literele din șirul s se incrementează frecvențele, apoi pentru literele din șirul t se decrementează frecvențele. La final se verifică dacă vectorul conține doar zerouri.
(g) Cu două liste de tupluri (literă, frecvență), se sortează listele, apoi se compară listele.
(h) Cu două seturi de tupluri (literă, frecvență), se compară seturile.
(i) Alte idei…?

2. În fișierul text “numar_lipsa.txt” se găsesc pe prima linie $n-1$ numere naturale distincte dintre primele n numere naturale nenule. Să se afișeze numărul lipsă.
   Exemplu: dacă fișierul “numar_lipsa.txt” conține “2 1 5 4”, se va afișa numărul lipsă 3.

```
f = open("numar_lipsa.txt", "r")
v = f.read()
s1 = {int(x) for x in v.split()}
s2 = {x for x in range(1, len(s1)+2)}
print("Numarul lipsa este:", *(s2-s1)) # diferenta seturi
f.close()
```

3. Fișierul text “numere_comune.txt” conține numere naturale despărțite prin spații și scrise pe mai multe linii. Să se afișeze în fișierul “comune.txt” numerele care apar pe toate liniile din fișier.
   Exemplu: dacă fișierul “numere_comune.txt” conține
   2 1 5 1 3
   1 4 2 2
   2 1 1 6 8
   atunci fișierul “comune.txt” va conține (numerele nu neapărat în această ordine):
   1 2

```
f = open("numere_comune.txt", "r") #pt citire
s = f.readline()
M1 = set([int(x) for x in s.split()])
while s != "": # sir vid cand s-a terminat fisierul
    s = f.readline()
    if s != "":
        M2 = set([int(x) for x in s.split()])
        M1 = M1 & M2 # intersectie de seturi
        # echivalent cu: M1.intersection(M2)
f.close()

f2 = open("comune.txt", "w") #pt scriere
f2.write(" ".join([str(x) for x in M1]))
f2.close()
```

Atenție, la metoda “**join**” elementele din colecția iterabilă primită ca parametru trebuie să fie toate de tip ***str***, altfel dă eroare.