Tutoriat 7 – Recapitulare
- rezolvări -

Subiectul I

Fișierul text `"teatru.in"` conține, pe mai multe linii, un fragment dintr-o piesă de teatru, respectiv pe fiecare linie se află câte o replică a unui personaj, sub forma `"personaj: replică"`. Numele unui personaj poate fi format din mai multe cuvinte, iar o replică nu va conține niciodată caracterul `':'`. Să se scrie în fișierul text `"teatru.out"` cuvintele din fragmentul dat grupate în funcție de numărul personajelor care le-au pronunțat, conform modelului din exemplul de mai jos. Cuvintele vor fi scrise în ordinea descrescătoare a numărului de personaje care le-au pronunțat, iar în caz de egalitate se vor scrie în ordine alfabetică. Pentru fiecare cuvânt, numele personajelor care l-au pronunțat vor fi ordonate alfabetic. Fiecare cuvânt va fi scris o singură dată și nu se va face distincție între litere mici și litere mari.

Exemplu:

teatru.in	teatru.out
Tipatescu: Misel! Pristanda: Curat misel! Tipatescu: Murdar! Pristanda: Curat murdar! Tipatescu: Ei! Nu s-alege! Pristanda: Nu s-alege! Farfuridi: Vrei sa vorbesc curat si deslusit, stimabile?	curat: Farfuridi,Pristanda misel: Pristanda,Tipatescu murdar: Pristanda,Tipatescu nu: Pristanda,Tipatescu s-alege: Pristanda,Tipatescu deslusit: Farfuridi ei: Tipatescu sa: Farfuridi si: Farfuridi stimabile: Farfuridi vorbesc: Farfuridi vrei: Farfuridi

```python
def elimina_semne( s ):
    return "".join( [ x for x in s.lower() if x.isalnum() or x == '-' ] )

with open( "teatru.in", 'r' ) as f:
    replici = f.readlines()

replici = [ x.split(':') for x in replici ]
replici = [ ( x[0], [ elimina_semne(y) for y in x[1].split() ] ) for x in replici ]

d = dict()
for replica in replici:
    for cuv in replica[1]:
        if not cuv in d.keys():
            d[cuv] = [ replica[0] ]
        else:
            d[cuv].append( replica[0] )
```

```
lista = list( zip( d.keys(), d.values() ) )
lista.sort( key = lambda x: ( -len( x[1] ), x ) )

with open( "teatru.out", 'w' ) as g:
    for tuplu in lista:
        personaje = list( set( tuplu[1] ) )
        personaje.sort()
        g.write( tuplu[0] + ': ' + ', '.join( personaje ) + '\n' )
```

Subiectul al II-lea

a) Fișierul `"date.in"` conține `n>1` linii cu următoarea structură: pe linia `i` se găsesc `n` numere naturale nenule separate prin câte un spațiu. Să se scrie o funcție `citire` care să citească datele din fișier și să returneze matricea de dimensiuni $n \times n$ care conține numerele în ordinea din fișier.

b) Să se scrie o funcție `modifica_matr` care primește ca parametri o matrice pătratică $n \times n$ și un număr variabil de parametri $x_1, x_2, \ldots x_k$ cu valori cuprinse între `0` și `n-1`, reprezentând indicii unor linii/coloane. Funcția va modifica matricea primită ca parametru astfel:

- adaugă o linie nouă la finalul matricei (după ultima linie existentă), în care fiecare element de pe coloana `j` va fi egal cu:
  - `-1`, dacă indicele `j` nu se află printre parametrii $x_1, x_2, \ldots x_k$ primiți de funcție sau
  - **suma** elementelor de pe coloana `j` aflate strict deasupra diagonalei principale, dacă indicele `j` se află printre parametrii $x_1, x_2, \ldots x_k$ primiți de funcție.
- apoi adaugă (la matricea obținută după adăugarea liniei) o coloană nouă la începutul matricei (înainte de prima coloană existentă), în care fiecare element de pe linia i va fi egal cu:
  - `-1`, dacă indicele `i` nu se află printre parametrii $x_1, x_2, \ldots x_k$ primiți de funcție sau
  - **maximul** dintre elementele de pe linia `i` aflate pe diagonala secundară sau sub ea, dacă indicele `i` se află printre parametrii $x_1, x_2, \ldots x_k$ primiți de funcție.

c) Să se apeleze funcția de la `b)` pentru matricea obținută la `a)` și indicii corespunzători următoarelor linii/coloane: prima, a doua, ultima, una respectiv două din mijlocul matricei (în funcție dacă n este impar respectiv par). Matricea obținută să se afișeze pe ecran, fără paranteze și virgule, iar elementele de pe fiecare coloană să fie aliniate la dreapta ținând cont că numerele pot avea maxim `4` caractere (inclusiv semnul `'-'`).

date.in	Afisare pe ecran
1 2 3 4 5 6 7 2 8 2 3 1 5 4 3 1 4 2 6 3 3 4 7 1 5 8 3 6 5 3 7 8 2 9 2 6 9 1 7 4 2 8 7 5 2 6 8 4 1	7 1 2 3 4 5 6 7 5 2 8 2 3 1 5 4 -1 3 1 4 2 6 3 3 8 4 7 1 5 8 3 6 -1 5 3 7 8 2 9 2 -1 6 9 1 7 4 2 8 8 7 5 2 6 8 4 1 -1 0 2 -1 9 -1 -1 30

Explicații: După modificări, se va obține matricea:

7	1	**2**	3	**4**	5	6	**7**
5	2	8	2	**3**	1	**5**	**4**
-1	3	1	4	**2**	6	3	**3**
8	4	7	1	**5**	**8**	**3**	**6**
-1	5	3	7	8	2	9	**2**
-1	6	9	1	7	4	2	**8**
8	**7**	**5**	**2**	**6**	**8**	**4**	**1**
-1	**0**	**2**	**-1**	**9**	**-1**	**-1**	**30**

```
def citire():
    with open( "date.in", 'r' ) as f:
        mat = f.readlines()
    mat = [ [ int(x) for x in linie.split() ] for linie in mat ]
    return mat

def modifica_matr( mat, *indici ):
    n = len( mat )
    mat.append( [ -1 if not j in indici else sum( [ mat[i][j] for i in range( j ) ]
) for j in range( n ) ] )
    mat[:] = [ [ -1 if not i in indici else max( mat[i][n - i - 1:] ) ] + mat[i]
for i in range( n + 1 ) ]

mat = citire()
n = len( mat )
modifica_matr( mat, 0, 1, n - 1, n // 2, (n - 1) // 2 )
print( '\n'.join( ''.join( [ str(x).rjust(4) for x in mat[i] ] ) for i in range( n
+ 1 ) ) )
```

Subiectul al III-lea

Fișierul text `"drumuri.in"` conține informații despre drumurile dintre orașele unei țări. O linie din fișier are următoarea structură:

`Nume_Oras_1 - Nume_Oras_2 distanta stare_drum`

unde `Nume_Oras_1` și `Nume_Oras_2` sunt numele a două orașe (un nume este un șir de cuvinte separate prin câte un spațiu), `distanta` este lungimea drumului dintre cele două orașe, iar `stare_drum` este un număr natural între `0` și `5` reprezentând calitatea drumului între cele două orașe. Pe un drum se poate circula doar într-un sens, respectiv de la `Nume_Oras_1` la `Nume_Oras_2`.

Un exemplu de astfel de fișier este:

drumuri.in

```
Oraselul Mic - Satul Mare 20 5
Oraselul Mic - Moeni 10 1
Satul Mare - Capitala 100 2
Satul Mare - Pol 20 5
Capitala - Pol 23 3
```

a) Să se memoreze datele din fișier într-o singură structură de date astfel încât să se **răspundă eficient** la cerințele de la punctele următoare (interogarea și modificarea informațiilor despre un drum, determinarea orașelor accesibile din alt oraș).
b) Scrieți o funcție `modifica_stare` care are următorii parametri (în această ordine):

- structura în care s-au memorat datele la cerința `a)`
- un număr natural `s` între `0` și `5` reprezentând starea unui drum
- două șiruri de caractere `o1` și `o2`; ultimul parametru `o2` are valoarea implicită șirul vid. Dacă `o2` este un șir nevid, funcția va modifica starea drumului de la orașul cu numele `o1` la orașul `o2` cu valoarea `s`, dacă acest drum există. Dacă `o2` este șirul vid funcția va modifica starea tuturor drumurilor de la orașul `o1` la celelalte orașe în `s`. Funcția va returna numărul de drumuri a căror stare a fost modificată.

c) Scrieți o funcție `accesibil` cu `2` parametri: structura în care s-au memorat datele la cerința `a)` și un număr variabil de șiruri de caractere reprezentând nume de orașe. Funcția returnează mulțimea orașelor la care se poate ajunge din cel puțin unul dintre orașele primite ca parametru folosind **unul** dintre drumurile din oraș. Apelați funcția pentru orașele `Oraselul Mic` și `Capitala` și afișați rezultatul obținut (ordinea orașelor din mulțimea returnată nu contează).

ieșire

```
{'Satul Mare', 'Moeni', 'Pol'}
```

```
def citire():
    global a
    with open("drumuri.in", 'r') as f:
        aux = f.readlines()
        for linie in aux:
            linie = linie.split("-")
            linie = [ x.split() for x in linie ]
            oras1 = " ".join( linie[0] )
            oras2 = " ".join( linie[1][:-2] )
            dist = int( linie[1][-2] )
            stare = int( linie[1][-1] )
            a[ (oras1, oras2) ] = ( dist, stare )

def modifica_stare( a, s, o1, o2 = "" ):
    cnt = 0
    if o2 != "" and (o1, o2) in a.keys():
        if a[ (o1, o2) ][1] != s:
            a[ (o1, o2) ] = ( a[ (o1, o2) ][0], s )
            cnt += 1
    else:
        chei = [ x for x in a.keys() if x[0] == o1 ]
        for cheie in chei:
            if a[cheie][1] != s:
                a[cheie] = ( a[cheie][0], s )
                cnt += 1
    return cnt

def accesibil( a, *orase ):
    chei = [ x for x in a.keys() for y in orase if x[0] == y ]
    rez = set()
    for cheie in chei:
        rez.add( cheie[1] )
    return rez

a = dict()
citire()
print( modifica_stare( a, 5, "Oraselul Mic" ) )
print( accesibil( a, "Oraselul Mic", "Capitala" ) )
```