Tutoriat 10 – Model Test Laborator 2
- rezolvări -

Subiectul I

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

with open("text.in", 'r') as f:
    cuv = faraSemne( f.readline().lower() )
    aux = [ faraSemne(x) for x in f.read().lower().split() ]

multime = set(cuv + '-')
d = dict()

for cuvant in aux:
    if set(cuvant) & multime == set(cuvant):
        aux = list( set(cuvant) - set('-') )
        aux.sort()
        cheie = tuple( aux )
        if cheie in d.keys():
            d[ cheie ].append(cuvant)
        else:
            d[ cheie ] = [ cuvant ]

chei = list( d.keys() )
chei.sort( key = lambda x: (-len(x), str(x)) )

with open("text.out", 'w') as g:
    for cheie in chei:
        g.write( f'Literele {list(cheie)}:\n' )
        d[cheie] = list( set( d[cheie] ) )
        d[cheie].sort( key = lambda x: (-len(x), x) )
        g.write( '\n'.join( d[cheie] ) )
        g.write('\n')

    if chei == []:
        g.write("Imposibil")
```

Subiectul II

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

def bordare_matrice( mat ):
    mat = mat[:]
    for linie in mat:
        linie.append( sum(linie) )
    mat.append( [ sum( [ mat[i][j] for i in range( len(mat) ) ] ) for j in range(
len(mat) + 1 ) ] )
    return mat

mat = bordare_matrice( citire_matrice() )
with open('matrice.out', 'w') as g:
    n = len(mat)
    for i in range( n ):
        g.write( f'{mat[i][i]} ' )
    for i in range( n ):
        if i != n - i - 1:
            g.write( f'{mat[i][n - i - 1]} ' )
    for i in range( n ):
        for j in range( n ):
            if i != j and i != n - j - 1:
                g.write( f'{mat[i][j]} ' )
```

Subiectul III

```
with open('legaturi.in', 'r') as f:
    aux = [ linie.split() for linie in f.readlines() ]

d = dict()
for linie in aux:
    p1 = tuple( [ int(x) for x in linie[0][1:-1].split(',') ] )
    p2 = tuple( [ int(x) for x in linie[1][1:-1].split(',') ] )
    culoare = linie[2]
    eticheta = ' '.join( linie[3:] )
    d[ (p1, p2) ] = ( culoare, eticheta )

def insereaza_legatura( d, p1, p2, culoare, eticheta ):
    if (p1, p2) in d.keys() or (p2, p1) in d.keys():
        return False
    d[ (p1, p2) ] = (culoare, eticheta)
    return True

insereaza_legatura( d, (1, 3), (2, 7), 'negru', 'legatura noua' )
for cheie in d.keys():
    print( list(cheie[0]), list(cheie[1]), d[cheie][0], d[cheie][1] )
```

```
def vecini( d, *tupluri ):
    v = []
    for tuplu in tupluri:
        vec = []
        for cheie in d.keys():
            if cheie[0] == tuplu:
                vec.append( cheie[1] )
            elif cheie[1] == tuplu:
                vec.append( cheie[0] )
        v.append( set(vec) )
    aux = v[0]
    for multime in v[1:]:
        aux = aux & multime
    aux = list(aux)
    aux.sort( key = lambda x: -x[1] )
    return aux

vec = vecini( d, (2, 7), (1, 2) )
print( vec )
```