**LABORATORUL 4**

3. Se citesc din fișierul “puncte.in” informații despre puncte în plan: cele două coordonate (numere întregi) și eticheta (care poate fi un șir ce conține și spatii). Mai exact, structura fișierului este următoarea: pe fiecare linie a fișierului sunt date informațiile despre un punct: doua coordonate întregi si eticheta punctului.

Asupra punctelor din fișier se fac operațiile codificate in fișierul “interogari.in” astfel: pe fiecare linie se dau doua numere reprezentând coordonatele unui punct si o valoare din mulțimea {1,0} cu semnificația interogare (1)/ștergere(0).

Ca rezultat al operațiilor se va crea fișierul interogari.out în care în dreptul punctelor cu 1 se va scrie eticheta (sau mesajul *nu exista*), iar la final se va afișa mesajul: “punctele ramase” și apoi vor afișa punctele rămase după ștergerea punctelor cu 0 (sub forma (x,y): eticheta). Dacă un punct apare de mai multe ori în puncte.in se păstrează ultima(variante:prima/toate) etichetă asociată lui.

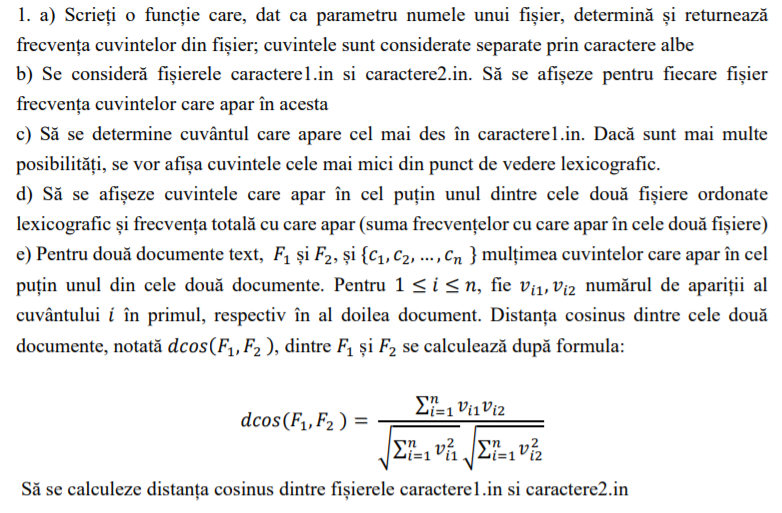
1 2 punctul 1  
1 3 punctul 2  
2 5 punctul 3  
1 2 punctul 1 nou  
4 1 punctul 4

f = open(**"puncte.in"**)  
d = {}  
**for** linie **in** f: *# x y eticheta\_cu\_space* ls = linie.split(maxsplit=2)  
 x = int(ls[0])  
 y = int(ls[1])  
  
 *# eticheta=" ".join(ls[2:]) - daca nu foloseam maxsplit* t = (x, y)  
 eticheta = ls[2].rstrip(**"\n"**)  
  
 *# varianta 1 - daca un pct se repeta ramane ultima eticheta* d[t] = eticheta  
  
 **'''   
  
 #varianta 2 - daca un pct se repeta ramane prima eticheta  
  
 if t not in d: #cautare dupa cheie   
 d[t]=eticheta   
   
 #varianta 2   
 d.setdefault(t,eticheta) #dc exista cheia nu modifica   
 '''** *# interogare:  
 # print(t,d.get(t, "nu exista"))  
 # print(d[t]) si t nu exista => eroare*print(d)

(1,2) punctul 1 nou  
(2,4) nu exista  
(4,1) punctul 4  
(10,20) nu exista  
punctele ramase  
(2, 5): punctul 3  
(4, 1): punctul 4

1 2 1  
2 4 1  
1 2 0  
1 3 0  
4 1 1  
10 20 1  
20 40 0

*#se afiseaza: {(1, 2): 'punctul 1 nou', (1, 3): 'punctul 2', (2, 5): 'punctul 3', (4, 1): 'punctul 4'}*f.close()  
f = open(**"interogari.in"**)  
g = open(**"interogari.out"**, **"w"**)  
**for** linie **in** f:  
 ls= linie.split()  
 x = int(ls[0])  
 y = int(ls[1])  
 t = (x, y)  
 **if** int(ls[2]) == 1:  
 **if** t **in** d:  
 g.write(**f"{**t**} {**d[t]**} \n"**)  
 **else**:  
 g.write(**f"{**t**} nu exista \n"**)  
 *# sau print(t,d.get(t,"nu exista"))* **if** int(ls[2]) == 0:  
 **if** t **in** d:  
 **del** d[t]  
 *# sau d.pop(t, "nu exista")*g.write(**"Punctele ramase\n"**)  
**for** t **in** d:  
 g.write(**f"{**t**} {**d[t]**} \n"**)  
*# sau for (t,eticheta) in d.items(): g.write(t,eticheta)*f.close()  
g.close()



**def** frecventa\_cuvinte(nume\_fisier):  
 f=open(nume\_fisier)  
 *#with open(nume\_fisier) as f:  
 #f.read() => tot fisierul  
  
 #linie=f.readline()  
 #while linie!="":*

d={}  
 **for** linie **in** f:  
 ls=linie.replace(**","**,**" "**).split()  
 **for** cuv **in** ls:  
 **if** cuv **in** d:  
 d[cuv]=d[cuv]+1  
 **else**:  
 d[cuv]=1  
 *#d[cuv]=d.get(cuv,0)+1* f.close()  
 **return** d  
*#b)*d1=frecventa\_cuvinte(**"fisier1.in"**)  
print(d1)  
d2=frecventa\_cuvinte(**"fisier2.in"**)  
print(d2)  
*#c)*frecv\_max=0  
cuv\_max=**""  
for** cuv **in** d1:  
 **if** d1[cuv]>frecv\_max:  
 frecv\_max=d1[cuv]  
 cuv\_max=cuv  
 **elif** d1[cuv]==frecv\_max:  
 **if** cuv<cuv\_max:  
 cuv\_max=cuv  
  
print(frecv\_max, cuv\_max)  
  
*#max(ls,key=cheie)*print(max(d1.values())) *#frecventa*print(d1.items())  
  
**def** cheie(x):  
 **return** x[1]  
  
print(max(d1.items(),key=cheie))  
t=max(d1.items(),key=cheie) *#tuplu (cheie,valoare)*print(t[0]) *#un cuvant cu frecventa maxima*

*#solutia 1***def** cheie(x):  
 **return** -x[1],x[0]  
  
print(min(d1.items(),key=cheie)[0])  
  
*#solutia 2:  
#def comparator(x,y): Tema(optional) cu return -1, 0, 1 si cu if-uri  
  
#d)  
#reuniunea cuvintelor din cele doua fisiere:  
  
#varianta 1 - 'manual'  
#ls=d1.keys()*ls=[]  
**for** cuv **in** d1:  
 ls.append(cuv)  
**for** cuv **in** d2:  
 **if** cuv **not in** d1:  
 ls.append(cuv)  
print(ls)  
  
*#varianta 2:*ls=d1.keys()|d2.keys()  
print(ls)  
  
**for** cuv **in** sorted(ls):  
 f1=0  
 **if** cuv **in** d1:  
 f1=d1[cuv]  
 f2 = 0  
 **if** cuv **in** d2:  
 f2 = d2[cuv]  
 print(**f"'{**cuv**}': {**f1+f2**}"**)  
 *#varianta 2- cu get* print(**f"'{**cuv**}': {**d1.get(cuv,0) + d2.get(cuv,0)**}"**)  
  
*#e)***from** math **import** sqrt  
prod\_scalar=0 *#numaratorul*norma\_1=0  
norma\_2=0  
**for** cuv **in** ls:  
 f1=d1.get(cuv, 0)  
 f2=d2.get(cuv, 0)  
 prod\_scalar=prod\_scalar+f1\*f2  
 norma\_1=norma\_1+f1\*\*2  
 norma\_2 = norma\_2 + f2\*\*2  
dcos=prod\_scalar/(sqrt(norma\_1)\*sqrt(norma\_2))  
print(dcos)  
print(**f"{**dcos**:.4f}"**)

2. În fișierul "elevi.in" sunt memorate informații despre elevii unei clase; astfel, pe o linie a fișierului se dau următoarele informații despre un elev: cnp, nume (fără spații), prenume (fără spații), lista de note (maxim 5).

 a)Memorați lista de elevi din fișier astfel încât să se poată răspundă cât mai eficient la întrebări de tipul: dat cnp elev, care sunt numele, notele.

b) ) Scrieți o funcție care primește ca parametri un cnp și informațiile despre elevi memorate la a) și crește cu 1 prima notă a elevului. Funcția returnează nota după modificare sau None dacă cnp-ul nu există. Apelați funcția pentru un cnp citit de la tastatură

c) Scrieți o funcție care primește ca parametri un cnp și informațiile despre elevi memorate la a) și șterge informațiile despre elevul cu acest cnp. Apelați funcția pentru un cnp citit de la tastatură (dacă cnp-ul nu este în listă funcția nu va modifica nimic și nu va da eroare)

d) Folosind memorarea de la a (nu citind din nou datele) construiți în memorie o lista cu perechi (nume, lista de note) cu elevii din fișier, ordonata după medie și, în caz de egalitate, după nume.

5234 Ion Ionica 10 8 2 9

1789 Mihail Marius 10 10 10

8923 Ionescu Maria 4 5 8 7

1789 Radulescu Matei 10 10

**Var facuta la lab:**

f=open(**"fisier.in"**)  
d={}  
**for** linie **in** f:  
 l=linie.split()  
 cod=int(l[0])  
 nume=l[1]  
 prenume=l[2]  
 note=[int(x) **for** x **in** l[3:]]  
 d[cod]=[nume, prenume, note]  
print(d)  
**def** crestere (cnp,d):  
 **if** cnp **in** d:  
 **if** d[cnp][2][0] < 10:  
 d[cnp][2][0]+=1  
 **return** d[cnp][2][0]  
 **return None**print(crestere(1789,d))  
print(crestere(8923,d))  
print(crestere(1769,d))  
print(d)  
val= d.values()  
print(val)  
**def** cheie(x):  
 suma=sum(x[2])  
 nr=len(x[2])  
 **return** (suma/nr, x[0], x[1])  
*# return (x[2][0],suma/nr, x[0], x[1]) mai intai ordonarea e dupa prima nota*sortare=sorted(val,key=cheie)  
print(sortare)  
**def** sterge (cnp,d):  
 d.pop(cnp,0)  
 **"""if cnp in d:  
 del d[cnp]  
 """**sterge(5234,d)  
sterge(5235,d) *#5235 nu exista*print(d)  
f.close()

**varianta completa:**

f=open(**"elevi.in"**)  
l=[]  
d=dict()  
line=f.readline()  
**while** line!=**""**:  
 cnp=**""** nume=**""** note=[]  
 nr=0  
 a=[]  
 **for** x **in** line.split():  
 *#print(x)* **if** nr==0:  
 cnp=x  
 **elif** nr==1:  
 nume=x  
 **elif** nr==2:  
 nume = nume + **" "** + x  
 **else**:  
 note.append(int(x))  
 nr=nr+1  
 a.append(nume)  
 a.append(note)  
 d[cnp]=a  
 line=f.readline()  
f.close()  
print(d)  
  
**def** fct(s,d):  
 **if not** s **in** d:  
 **return None  
 else**:  
 *#print(d[s][1])* nota=d[s][1][0]+1  
 **return** nota  
**def** sterge(s,d):  
 **if not** s **in** d:  
 **return** d  
 **else**:  
 **del** d[s]  
 **return** d  
s=input()  
print(fct(s,d))  
print(sterge(s,d))  
lista=[]  
*#print(d)***for** x **in** d:  
 nume=d[x][0]  
 note=d[x][1]  
 nr = 0  
 sum = 0  
 **for** y **in** note:  
 nr = nr + 1  
 sum = sum + y  
 med = sum / nr  
 li=[]  
 li.append(nume)  
 li.append(note)  
 li.append(med)  
 lista.append(li)  
print(lista)  
**for** i **in** range(len(lista)):  
 **for** j **in** range(i+1,len(lista)):  
 **if** lista[i][2]>lista[j][2]:  
 lista[i],lista[j]=lista[j],lista[i]  
 **else**:  
 **if** lista[i][2] == lista[j][2]:  
 **if** lista[i][0]>lista[j][0]:  
 lista[i], lista[j] = lista[j], lista[i]  
print(lista)

3. În fișierul text “test.in” se află testul unui elev de clasa a II-a la matematică, conținând 9 înmulțiri scrise pe rânduri distincte. Un calcul corect este notat cu un punct, iar unul incorect cu 0 puncte. Să se realizeze un program care să evalueze testul dat, astfel: în dreptul fiecărui calcul corect se va scrie mesajul ‘Corect’, iar în dreptul fiecărui calcul greșit se va scrie mesajul ‘Gresit’ și rezultatul corect, iar la final se va scrie nota (un punct se acordă din oficiu!). Rezultatul evaluării testului se va scrie în fișierul text “test.out”.

|  |  |
| --- | --- |
| test.in  3\*4=11  2\*10=20  5\*5=24  7\*4=28  2\*6=12  10\*10=100  3\*9=27  6\*7=33  0\*9=1 | test.out  3\*4=11 Gresit 12  2\*10=20 Corect  5\*5=24 Gresit 25  7\*4=28 Corect  2\*6=12 Corect  10\*10=100 Corect  3\*9=27 Corect  6\*7=33 Gresit 42  0\*9=1 Gresit 0  Nota 6 |

f=open(**"test.in"**)  
g=open(**"test.out"**, **"w"**)  
nota=0  
**for** linie **in** f:  
 a,b=linie.split(**"\*"**) *#b va ramane ceva de genul 4=11 pentru prima linie* c,d=b.split(**"="**)  
 a=int(a) *#primul termen din inmultire* c=int(c) *#al doilea* d=int(d) *#rezultatul* **if** a\*c==d:  
 nota+=1  
 g.write(**f"{**a**}\*{**c**}={**d**} Corect"**)  
 g.write(**"\n"**)  
 **else**:  
 g.write(**f"{**a**}\*{**c**}={**d**} Gresit {**a\*c**}"**)  
 g.write(**"\n"**)  
g.write(**f"Nota {**nota+1**}"**)  
g.close()

4. Scrieți un program care să determine grupurile de cuvinte dintr-un fișier text care p-rimează între ele = au aceleași ultime p-litere (p citit de la tastatura). Numele fișierului de intrare se va citi de la tastatură, iar grupurile formate din cel puțin două cuvinte distincte se vor scrie în fișierul text

“rime.txt”, câte un grup pe o linie. Cuvintele din fiecare grup vor fi sortate alfabetic.

f=open(input())  
g=open(**"rime.txt"**,**"w"**)  
d={}  
nr=0  
p=int(input())  
**for** linie **in** f:  
 **for** cuv **in** linie.split():  
 lungime=len(cuv)  
 rima=cuv[lungime-p:lungime]  
  
 **if** rima **in** d:  
 d[rima].append(cuv)  
  
 **else**:  
 lista=[]  
 lista.append(cuv)  
 d[rima]=lista  
  
**for** cuv **in** d:  
 list=[]  
 list=d[cuv]  
 l=sorted(list, reverse=**False**)  
 d[cuv]=l  
  
**for** rima,lista\_cuvinte **in** d.items():  
 **if** len(lista\_cuvinte)>=2:  
 g.write(str(lista\_cuvinte))  
 g.write(**"\n"**)  
f.close()  
g.close()

5. În fișierul text “date.in” sunt memorate, pe linii, numele și prenumele studenților dintr-o grupă. Să se scrie un program care să genereze conturile de email ale studenților și parolele temporare, după care să le salveze în fișierul text “date.out”. Contul de email al unui student va fi de forma prenume.nume@s.unibuc.ro, iar parola temporară va fi de forma o literă mare, 3 litere mici și 4 cifre.

Pentru genearerea de numere/litere aleatoare – modulul random (metodele randint,choice)

*date.in*

Bobocea Andrei

Marinescu Ciprian

Vasile Dragos

*date.out*

andrei.bobocea@s.unibuc.ro,Wadf2133

ciprian.marinescu@s.unibuc.ro,Qsdd2111

dragos.vasile@s.unibuc.ro,Bbyt7690

f=open(**"date.in"**)  
g=open(**"date.out"**, **"w"**)  
**import** random *#pt a genera numere aleatorii***import** string *#pt a genera litere aleatorii*s=**"@s.unibuc.ro"**litere=string.ascii\_letters *#abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ*litere\_mici=litere[:len(litere)//2]  
litere\_mari=litere[len(litere)//2+1:]  
cifre=string.digits *#0123456789***for** linie **in** f:  
 nume, prenume=linie.split()  
 nume=nume.lower()  
 prenume=prenume.lower()  
 c1 = random.choice(litere\_mari)  
 mici = **""**.join(random.choices(litere\_mici, k=3))  
 numere = **""**.join(random.choices(cifre, k=4))  
 parola = c1 + mici + numere  
 mail = nume + **"."** + prenume + s + **","** + parola  
 g.write(mail)  
 g.write(**"\n"**)  
f.close()  
g.close()

6.Scrieți o funcție “negative\_pozitive” care primește ca parametru o lista de numere întregi și returnează două liste, prima formată din numerele strict negative din lista primită ca parametru și a doua formată din cele strict pozitive. Scrieți un program care citește de la tastatură numele unui fișier text și apoi, folosind apeluri utile ale funcției “negative\_pozitive” scrie la sfârșitul fișierului text, pe două linii separate, numerele strict negative, respectiv cele strict pozitive, ordonate crescător.

**def** negative\_pozitive(list):  
 n=[]  
 p=[]  
 **for** x **in** list:  
 **if** x>0:  
 p.append(x)  
 **elif** x<0:  
 n.append(x)  
 **return** p,n  
  
g=open(input(), **"w"**) *#citirea numelui unui fisier de la tastatura, fisier in care se va scrie ceva*s=input()  
l=[int(x) **for** x **in** s.split()]  
p,n=negative\_pozitive(l)  
p.sort()  
n.sort()  
g.write(str(n))  
g.write(**"\n"**)  
g.write(str(p))  
g.close()