Programarea Algoritmilor
– LABORATOR NR. 3 –

Liste, tupluri, seturi, dicționare; Fișiere text

1. **TEMĂ** 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 de tip CSV “date.out”. Contul de email al unui student va fi de forma prenume.nume@myfmi.unibuc.ro, iar parola temporară va fi de forma o literă mare, 3 litere mici și 4 cifre.

date.in	*date.out*
Bobocea Andrei	andrei.bobocea@myfmi.unibuc.ro,Wadf2133
Marinescu Ciprian	ciprian.marinescu@myfmi.unibuc.ro,Qsdd2111
Vasile Dragos	dragos.vasile@myfmi.unibuc.ro,Bbyt7690

2. 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”. **Exemplu:**

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

3. Total cheltuieli

Un fișier text conține o pagină din jurnalul Anei, în care ea și-a notat cheltuielile efectuate într-o anumită zi. Scrieți un program care să afișeze suma totală cheltuită de Ana în ziua respectivă.

Exemplu: fișierul “cheltuieli.txt” conține textul:
“Azi am cumpărat 5.5 kg de mere cu 2.2 RON kilogramul și 3 pâini a câte 5 RON fiecare. ”

Observații:

- Se garantează faptul că există un număr par de numere pozitive întregi/reale în text, fiecare pereche reprezentând cantitatea și prețul (nu neapărat în această ordine) pentru un produs.
- Orice număr este delimitat de spații. După ce împărțiți textul în cuvinte, pentru a verifica dacă un cuvânt `cuv` este un număr întreg folosiți metoda `isdecimal()`, care returnează `True` doar dacă toate caracterele din `cuv` sunt cifre în baza 10. Pentru a verifica dacă un cuvânt este un număr real, “spargeți” cuvântul după “.” folosind metoda `split()`, iar dacă obțineți *exact două* subșiruri, verificați pentru fiecare dintre ele dacă este număr natural, folosind metoda `isdecimal()`.

4. Se citește de la tastatură în șir de numere întregi (tastați spațiu între numere și Enter doar la finalul șirului). Folosiți “list comprehensions” pentru a salva numerele într-o listă. Să se determine cele mai mari două valori distincte din șir. Atenție la cazul particular când șirul nu conține cel puțin două valori distincte!

5. Să se determine cuvântul care apare cel mai des într-o propoziție dată, precum și cuvântul care apare cel mai rar. Dacă sunt mai multe posibilități, se vor afișa cuvintele cele mai mici din punct de vedere lexicografic. **Sugestie de rezolvare:** se creează mai întâi un dicționar în care cheile sunt cuvintele și valorile asociate cheilor sunt frecvențele lor, după care se creează un alt dicționar în care frecvențele sunt chei și valorile asociate sunt liste formate din cuvintele având frecvența respectivă, după care se determină minimul/maximul cheilor din noul dicționar și se afișează șirul minim lexicografic din lista asociată minimului/maximului.

6. **TEMĂ** Folosind un dicționar, să se numere de câte ori apare fiecare caracter într-un text de tip "lorem ipsum". Folosiți generatorul de text https://loremipsum.io/.

7. **TEMĂ** Să se unifice două dicționare în care cheile sunt șiruri de caractere, iar valorile sunt de tip numeric. Astfel, în rezultat va apărea fiecare cheie distinctă, iar pentru o cheie care apare în ambele dicționare inițiale valoarea corespunzătoare va fi egală cu suma valorilor asociate cheii respective în cele două dicționare.

8. **TEMĂ** Dintr-un fișier text se citește un șir de caractere (o frază), iar de la tastatură se citește un cuvânt. Scrieți un program care să furnizeze toate cuvintele din frază formate din aceleași litere cu ale cuvântului dat (fără a fi neapărat anagrame!). Dacă șirul nu conține nici un cuvânt cu proprietatea cerută, programul trebuie să afișeze un mesaj corespunzător. Cuvintele din șir sunt despărțite între ele prin spații și semnele de punctuație uzuale. De exemplu, pentru șirul “Langa o cabana, stand pe o banca, un bacan a spus un banc bun.” și cuvântul “bacan” funcția trebuie să furnizeze cuvintele “cabana”, “banca”, “bacan” și “banc”.

9. Scrieți un program care să determine grupurile de cuvinte dintr-un fișier text care rimează între ele. 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.

10. Magazine și produse *(vedeți și: Seminar 3 (grupa 131), problema 3)*

Fișierul text “inventar.txt” conține pe fiecare rând, cu spațiu între ele, un șir de litere reprezentând numele unui magazin, urmat de numere naturale reprezentând codurile produselor aflate în stoc în magazinul respectiv.

a) Memorați informațiile citite din fișier într-un dicționar având pe post de chei numele magazinelor, iar ca valori mulțimi formată din codurile produselor din acel magazin;

b) Afișați codurile acelor produse care există în stocul tuturor magazinelor *(intersecție de mulțimi)*;

c) Afișați codurile tuturor produselor existente în toate magazinele *(reuniune de mulțimi)*;

d) Afișați pentru fiecare magazin: numele magazinului și codurile produselor exclusiviste (produse care se află doar în acel magazin) *(diferență de mulțimi)*.

11. Definiții

Un student vrea, pentru ora de engleză de la FMI, să găsească cele mai expresive cuvinte. Pentru aceasta a făcut o poză la astfel de cuvinte dintr-un dicționar, iar apoi a folosit un program ca să extragă un șir de caractere cu textul din poze. Textul respecta regulile:

- este împărțit în paragrafe, fiecare paragraf terminându-se cu "\n";
- fiecare paragraf corespunde unei intrări din dicționar, respectiv conține un cuvânt și definițiile sale;
- cuvântul asociat unui paragraf se află la începutul său, fiind urmat de ":" și apoi de toate definițiile sale.

Acum, studentul vrea sa decidă care sunt cele mai expresive cuvinte. În acest scop, el numără pentru fiecare cuvânt în câte expresii apare (Ex: **run** away, **run** over, **run** a company), astfel:

- numără de câte ori apare cuvântul în paragraf;
- numără de câte ori apare caracterul tilda (~) în paragraf;
- însumează cele două numere obținute.

Scrieți un program care să citească dintr-un fișier text șirul de caractere extras și contruiește o listă de tupluri formate dintr-un cuvânt aflat la început de paragraf și numărul care reprezintă expresivitatea sa calculată în acel paragraf.

De exemplu, pentru textul

"run: to go faster than a walk : to go steadily by springing steps : to take part into a contest - **~** a marathon : to move at a fast gallop - he may occasionally **run** to and from work : flee, retreat, escape - drop the gun and **run** : to go without restraint : move freely about at will - let chickens **~** loose : consort - we **run** with our group \n" +
"dog: canid wolves, foxes, and other dogs especially : a highly variable domestic mammal : a pet **~** : fellow, chap, a lazy person - you lucky **dog** \n" +
"break: **break** a/the record to do something better than the best known speed, time, number, etc. previously achieved : to fail to keep a law, rule, or promise = **~** the law : These enzymes **break** down food in the stomach (= cause food to separate into smaller pieces). I needed something to **break** the monotony of my typing job. The phone rang, as to **break** my concentration. To **~** (of a storm) = to start suddenly: We arrived just as a storm was breaking. \n"

se va obține:

[("run", 5), ("dog", 2), ("break", 6)]
de exemplu, pentru "run" numărăm de două ori pe **~** și de 3 ori pe **run,** deci expresivitatea este 5, iar tuplul este ("run", 5)

12. Departajare

La facultate se ține un concurs de programare. Organizatorii au aranjat ca punctajele să fie calculate automat, din codul scris. Totuși, ei nu au implementat vreo metodă de a departaja punctajele egale, așa că în ultimul moment s-au decis ca persoana care a trimis codul prima să fie clasată mai sus decât cel de-al doilea ca timp, dar cu punctaj identic, ș.a.m.d.

Până la introducerea lui “-1”, să se citească de la tastatură perechi de două elemente: (punctaj, nume_student), cu punctajul între 0 și 100. Perechile se consideră a fi introduse în ordinea predării codului de către participanți.
a) Salvați datele de intrare într-o listă de tupluri (punctaj, nume_student, nr_ordine), astfel încât să se poată deduce și numărul de ordine pentru fiecare participant (al câtelea a predat codul).
b) Să se afișeze lista tuturor punctajelor distincte obținute de participanți (folosiți un set).
c) Într-un dicționar, pentru fiecare punctaj să se asocieze o listă de tupluri ce conțin numele participantului și numărul său de ordine. Apoi, să se afișeze clasamentul concursului (descrescător după punctaje, pe câte un rând: punctaj, nume_participant și nr_ordine).

De exemplu, pentru următorul input:

64 Danil Marius
70 Derek Alexandru
100 Pirpiric Claudiu
18 Alexandrescu Matias
64 Popescu Catalin
100 Cozia Daniel
82 Stefan Dinca
-1

Vom obtine:

a) Lista [(64, Danil Marius, 1), (70, Derek Alexandru, 2), (100, Pirpiric Claudiu, 3), (18, Alexandrescu Matias, 4), (64, Popescu Catalin, 5), (100, Cozia Daniel, 6), (82, Stefan Dinca, 7)]

b) Mulțimea {100, 82, 70, 64, 18} (cu elementele nu neapărat în această ordine)

c) Și dicționarul (având perechile nu neapărat în această ordine):

```
{
  100: [("Pirpiric Claudiu", 3), ("Cozia Daniel", 6)],
   82: [("Dinca Stefan", 7)],
   70: [("Derek Alexandru", 2)],
   64: [("Danil Marius", 1), ("Popescu Catalin", 5)],
   18: [("Alexandrescu Matias", 4)]
}
```

13. Se citește un număr natural N.

a) Să se genereze și afișeze o matrice de dimensiune NxN, cu elementele de la 1 la N*N - în ordine crescătoare, de la stânga la dreapta și de sus în jos.

b) Pentru a parcurge elementele matricei în spirală, pornind din colțul din stânga-sus (spre dreapta, în jos, spre stânga, în sus, …), să se obțină întâi o listă având elemente de tip tuplu (linie, coloană) care să reprezinte pozițiile care trebuie parcurse în această spirală.

c) Folosind lista de tupluri de mai sus, să se afișeze elementele din matrice aflate la acele poziții.

L\C	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

```
lista_poz = [(0, 0), (0, 1), ..., (0, N-2),
             (0, N-1), (1, N-1), ..., (N-2, N-1),
             (N-1, N-1), (N-1, N-2), ..., (N-1, 1),
             (N-1, 0), (N-2, 0), ..., (1, 0),
             (1, 1), (1, 2), ...]
```

Pentru N = 5:

```
spirala = [1, 2, 3, 4, 5, 10, 15, 20, 25, 24, 23, 22,
21, 16, 11, 6, 7, 8, 9, 14, 19, 18, 17, 12, 13]
```

14. Se citește din fișierul “graf_in.txt” un graf (ne)orientat astfel:

- cuvântul “orientat” sau “neorientat”
- n - numărul de noduri (cu nodurile indexate de la 1 la n)
- m - numărul de arce (muchii)
- m perechi de forma (xi,yi) reprezentând lista de arce (muchii)
- două noduri s și f (start și final)

Pentru toate cerințele faceți afișările într-un fișier “graf_out.txt”.
Nu uitați să închideți fișierele la final!

a) Să se afișeze *lista de arce (muchii)*: o listă de tupluri cu elemente (i,j) pentru arc de la nodul i la nodul j, pt. graf orientat, respectiv pentru muchie între nodurile i și j, pt. graf neorientat.
b) Să se genereze *listele de adiacență* pentru graf: folosiți un dicționar D în care cheia este nodul curent i, iar valoarea este o listă de noduri j, adiacente cu nodul curent (există arc de la nodul i la nodul j, pt. graf orientat, respectiv există muchie între nodurile i și j, pt graf neorientat).
c) Să se genereze *matricea de adiacență* a grafului: o matrice M de dimensiune n*n, în care M[i][j] = 1 dacă există arc de la nodul i la nodul j (pt. graf orientat) sau M[i][j] = M[j][i] = 1 dacă există muchie între nodurile i și j (pt. graf neorientat), și valoarea 0 în restul matricei.

d) Să se obțină *parcurgerea în lățime (breadth first*) pornind din nodul s.
1) Se ia o listă BF, inițial vidă (`BF=[]`), care se va comporta ca o *coadă* și va avea ca elemente tupluri (nod, tata_nod). Nu se șterg elementele introduse în lista BF, dar se iau două variabile st(ânga) și dr(eapta) care vor reține poziția curentă a primului, respectiv ultimului element al cozii. Spunem despre nodurile aflate în lista BF că sunt "vizitate" deja.
2) Se adaugă primul element (tuplul format din nodul s și -1 pt. tatăl lui inexistent) în coadă (adică la finalul listei BF): `BF.append((s,-1)); st=dr=0`.
3) Cât timp coada nu este vidă (`while st<=dr:`) se execută:
→ Pentru primul nod din coadă (`tata=BF[st][0]`), se găsesc toți vecinii (din dicționarul obținut la punctul b) (`for fiul in D[tata]:`) nevizitați (care nu se află în lista BF) (`if fiul not in [nod for (nod, parinte) in BF]:`) și se adaugă la finalul cozii împreună cu părintele lor (`BF.append((fiul,tata)); dr+=1`).
→ Apoi se "elimină" primul nod din coadă (`st+=1`) și se reia pasul 3).
4) Să se afișeze parcurgerea în lățime: doar primul element al fiecărui tuplu din lista BF.

e) Să se obțină *parcurgerea în adâncime (depth first)* pornind din nodul s.
1) Se ia o listă DF, inițial vidă (`DF=[]`), care se va comporta ca o *stivă* și va avea ca elemente noduri ale grafului; și o listă "vizitat", inițial vidă (`vizitat=[]`), în care se vor adăuga nodurile pe măsură ce sunt vizitate în cadrul parcurgerii în adâncime.
2) Se adaugă în stivă primul element (nodul s) și se marchează ca fiind vizitat (`DF.append(s); vizitat.append(s)`).
3) Cât timp stiva nu este vidă (`while DF!=[]:`) se execută:
→ pentru nodul din vârful stivei (ultimul element al listei DF) (`tata=DF[-1]`) se caută în dicționarul D un vecin (`for fiul in D[tata]:`) nevizitat (`if fiul not vizitat: break`).
i) dacă acest fiu există, atunci acesta se adaugă în vârful stivei (la finalul listei DF) și se vizitează (`DF.append(fiul); vizitat.append(fiul)`), apoi se reia pasul 3).
ii) dacă acest fiu nu există (nodul nu are fii sau toți fiii sunt deja vizitați), atunci se scoate nodul din vârful stivei (de la finalul listei DF) (`DF.pop(-1)`), apoi se reia pasul 3).
4) Să se afișeze parcurgerea în adâncime: lista "vizitat".

f) Să se găsească drumul de lungime minimă între nodurile s și f, dacă acesta există. În caz că nu există drum, să se afișeze un mesaj corespunzător.
Indicație de rezolvare: Folosind lista BF (obținută la punctul d), obțineți nodurile care formează drumul, plecând din nodul f și trecând din tată în tată până ajungeți la nodul s (atenție la ordinea în care afișați nodurile drumului, trebuie de la s către f).

15. Fișierul text “persoane.in” conține, pe fiecare linie, informații despre câte o persoană, în forma indicată mai jos. Să se citească informațiile din fișier și să creeze o listă de dicționare în care informațiile referitoare la o persoană să fie memorate într-un dicționar având cheile principale nume, prenume, adresa și cheile secundare oraș, stradă, număr. De asemenea, să se creeze un dicționar care să permită afișarea tuturor persoanelor care locuiesc într-un anumit oraș.

Exemplu: “*persoane.in*” conține:

nume:Popescu,prenume:Ion,adresa:{oras:Bucuresti,strada:Iuliu Maniu,numar:6}
prenume:Ana,nume:Grigore,adresa:{oras:Ploiesti,strada:Sperantei,numar:9}
nume:Mihai,prenume:Dan,adresa:{strada:Silozului,numar:10,oras:Bucuresti }