1) a. Scrieți o funcție min\_max care primește ca parametru o listă nevidă de numere întregi și returneză două valori: valoarea minimă și valoarea maximă din lista primită ca parametru. De exemplu, pentru lista [3, -3, 1, 7, 3, 2] funcția trebuie să returneze valorile -3 și 7. (1 p.)

def min\_max(L):  
 return min(L), max(L)

b. Considerăm un fișier text care conține pe fiecare linie câte un șir format din numere întregi. Scrieți o funcție incarca\_fisier care să încarce conținutul unui astfel de fișier text într-o listă, astfel: fiecare element al listei va fi o listă conținând toate numerele de pe o linie a fișierului text. Funcția va primi ca parametru numele fișierului text și va returna lista creată. De exemplu, daca fișierul text este 1 210 7 2 2 2 -3 1 -2 1 9 210 -30 210 10 atunci funcția trebuie sa furnizeze lista [[1, 210, 7], [2, 2, 2], [-3, 1, -2, 1], [9], [210, -30, 210, 10]]. Fiecare linie conține cel puțin un număr întreg, iar numerele de pe o linie sunt despărțite între ele prin unul sau mai multe spații.

def incarca\_fisier(f):  
 L = []  
 for linie in f:  
 linie1 = [int(element) for element in linie.split(' ')]  
 L.append(linie1)  
 return L

c. Folosind apeluri utile ale funcțiilor definite anterior, scrieți un program care să citească de la tastatură numele unui fișier text de tipul descris mai sus și apoi să rezolve următoarele două cerințe:

• Să scrie în fișierul text egale.txt numerele de ordine ale liniilor din fișierul respectiv care conțin doar valori egale între ele. În cazul în care nu există nicio linie cu proprietatea cerută, se va scrie în fișierul egale.txt mesajul "Nu există!". Liniile fișierului text de intrare se consideră numerotate de la 0, de sus în jos. De exemplu, pentru fișierul text de mai sus, în fișierul rezultat.txt trebuie scrise, pe două linii, numerele 1 și 3.

• Să afișeze pe ecran două numere întregi a și b din fișierul text dat cu proprietatea că toate numerele din fișierul text aparțin intervalului [a, b]. De exemplu, pentru fișierul text de mai sus, trebuie afișate numerele a = -30 și b = 210.

def min\_max(L):  
 return min(L), max(L)  
  
def incarca\_fisier(f):  
 L = []  
 for linie in f:  
 linie1 = [int(element) for element in linie.split(' ')]  
 L.append(linie1)  
 return L  
  
f = open("egale.txt")  
L = incarca\_fisier(f)  
nr = -1  
n = 0  
for linie in L:  
 nr += 1  
 x,y = min\_max(linie)  
 if x == y:  
 g = open("egale.txt", "a")  
 g.write("\n" + str(nr))  
 g.close()  
 n = 1  
if n == 0:  
 g = open("egale.txt", "a")  
 g.write("\nNu exista!")  
 g.close()  
  
M = []  
for linie in L:  
 for element in linie:  
 M.append(element)  
a,b = min\_max(M)  
print("a =", a, end=' si ')  
print("b =", b)  
f.close()

2) a) Scrieți o funcție deviruseaza care primește ca parametru o propoziție modificată de un virus și returnează propoziția corectată: virusul a acționat asupra propoziției inițiale (formată din cuvinte separate prin câte un spațiu) astfel: la fiecare cuvânt a interschimbat prima literă cu ultima, apoi inversat ordinea în care cuvintele apar în propoziție (primul a devenit ultimul, a doilea penultimul etc). De exemplu, pentru propoziția “aorectc aropozitip este aceasta” primita ca parametru funcția va returna “aceasta este propozitia corecta”

def deviruseaza(prop):  
 L = []  
 prop = prop.split()[::-1]  
 for cuvant in prop:  
 a = cuvant[0]  
 b = cuvant[-1]  
 cuvant = b + cuvant[1:]  
 cuvant = cuvant[:-1] + a  
 L.append(cuvant)  
 return " ".join(L)

b) Scrieți o funcție prime care primește ca parametru un număr natural n și un parametru numar\_maxim cu valoarea implicită 0 care returnează o listă cu numere prime mai mici decât n. Dacă se specifică o valoare nenulă pentru parametrul numar\_maxim, atunci funcția va returna doar primele numar\_maxim numere prime mai mici decât n.

def prime(n, numar\_maxim):  
 nr = 0  
 nr\_prime = []  
 for i in range(2, n+1):  
 d=0  
 j=2  
 while j \* j <= i:  
 if i%j==0:  
 d += 1  
 j += 1  
 if d == 0:  
 if numar\_maxim == 0:  
 nr\_prime.append(i)  
 else:  
 nr += 1  
 if nr <= numar\_maxim:  
 nr\_prime.append(i)  
 return nr\_prime

c) Se dă fișierul intrare.in în care fiecare linie reprezintă o propoziție în care cuvintele sunt separate prin câte un spațiu. Propozițiile din acest fișier au fost modificate de un virus care acționează așa cum a fost descris la punctul a), dar care a acționat doar asupra liniilor de indice prim din fișier (numerotarea liniilor începând de la 1). Folosind apeluri ale funcțiilor de la a) si b) să se creeze fișierul intrare\_devirusata.out cu propozițiile din fișierul inițial, dar cu cele de pe poziții prime corectate.

def deviruseaza(prop):  
 L = []  
 prop = prop.split()[::-1]  
 for cuvant in prop:  
 a = cuvant[0]  
 b = cuvant[-1]  
 cuvant = b + cuvant[1:]  
 cuvant = cuvant[:-1] + a  
 L.append(cuvant)  
 return " ".join(L)  
  
def prime(n, numar\_maxim):  
 nr = 0  
 nr\_prime = []  
 for i in range(2, n+1):  
 d=0  
 j=2  
 while j \* j <= i:  
 if i%j==0:  
 d += 1  
 j += 1  
 if d == 0:  
 if numar\_maxim == 0:  
 nr\_prime.append(i)  
 else:  
 nr += 1  
 if nr <= numar\_maxim:  
 nr\_prime.append(i)  
 return nr\_prime  
  
f = open("intrare.in")  
nr = 0  
L = []  
for linie in f:  
 nr += 1  
 if nr == 1:  
 linie = linie.split()  
 linie = ' '.join(linie)  
 L.append(linie)  
 else:  
 if nr in prime(nr+1, 0):  
 L.append(deviruseaza(linie))  
 else:  
 linie = linie.split()  
 linie = ' '.join(linie)  
 L.append(linie)  
g = open("intrare\_devirusata.out", "w")  
g.write("\n".join(L))  
g.close()  
f.close()

3 a) Scrieți o funcție cifra\_control care primește ca parametru un număr natural n și returnează cifra de control a numărului n. Cifra de control a unui număr natural se obține efectuând suma cifrelor sale, apoi suma cifrelor acestei sume, până când suma obținută este un număr format dintr-o singură cifră.

def cifra\_control(n):  
 if n%9 == 0:  
 return 9  
 else:  
 return n%9

b) Scrieți o funcție insereaza\_cifra\_control care primește ca parametru o listă de numere naturale și o modifică inserând după fiecare element din listă cifra sa de control (!funcția modifică lista primita ca parametru, nu returnează lista modificată). Cifra de control a unui număr se va calcula folosind funcția de la a).

def insereaza\_cifra\_control(L):  
 for i in range(0, len(L)\*2, 2):  
 X = []  
 x = cifra\_control(L[i])  
 X.append(x)  
 L[i+1:i+1] = X

c) Scrieți o funcție egale care primește ca parametri un număr variabil de liste cu elemente numere naturale și întoarce True dacă listele primite ca parametru sunt egale și False altfel.

def egale(\*args):  
 L = []  
 nr = 0  
 for arg in args:  
 L.append(arg)  
 for i in range(1,len(L)):  
 if len(L[0]) == len(L[i]):  
 for j in range(len(L[0])):  
 if L[0][j] == L[i][j]:  
 nr += 1  
 if nr == len(L[0]):  
 return True  
 else:  
 return False

d) Se consideră fișierul numere.in care conține numere naturale separate prin spații. Folosind funcția de la b) afișați pe ecran fiecare număr din fișier urmat de cifra sa de control. După fiecare cifră de control afișată se va trece la linie nouă.

def cifra\_control(n):  
 if n%9 == 0:  
 return 9  
 else:  
 return n%9  
  
def insereaza\_cifra\_control(L):  
 for i in range(0, len(L)\*2, 2):  
 X = []  
 x = cifra\_control(L[i])  
 X.append(x)  
 L[i+1:i+1] = X  
  
def egale(\*args):  
 L = []  
 nr = 0  
 for arg in args:  
 L.append(arg)  
 for i in range(1,len(L)):  
 if len(L[0]) == len(L[i]):  
 for j in range(len(L[0])):  
 if L[0][j] == L[i][j]:  
 nr += 1  
 if nr == len(L[0]):  
 return True  
 else:  
 return False  
  
f = open("numere.in")  
x = [int(i) for i in (f.read()).split()]  
insereaza\_cifra\_control(x)  
for i in range(0, len(x), 2):  
 print(x[i], x[i+1])  
f.close()

e) Se dau două fișiere numere.in și numere2.in conținând numere naturale separate prin spații. Asociem fiecărui fișier un șir de cifre de control astfel: considerăm numerele distincte din fișier în ordine crescătoare și formăm un șir cu cifrele lor de control (în această ordine). Spunem că cele două fișiere sunt similare dacă șirurile asociate celor două fișiere sunt egale. Folosind funcțiile de la punctele b și c să se determine dacă cele două fișiere sunt similare si să se afișeze pe ecran un mesaj corespunzător da/nu.

def cifra\_control(n):  
 if n%9 == 0:  
 return 9  
 else:  
 return n%9  
  
def insereaza\_cifra\_control(L):  
 for i in range(0, len(L)\*2, 2):  
 X = []  
 x = cifra\_control(L[i])  
 X.append(x)  
 L[i+1:i+1] = X  
  
def egale(\*args):  
 L = []  
 nr = 0  
 for arg in args:  
 L.append(arg)  
 for i in range(1,len(L)):  
 if len(L[0]) == len(L[i]):  
 for j in range(len(L[0])):  
 if L[0][j] == L[i][j]:  
 nr += 1  
 if nr == len(L[0]):  
 return True  
 else:  
 return False  
  
f = open("numere.in")  
g = open("numere2.in")  
  
F = f.read()  
G = g.read()  
  
F = [int(i) for i in F.split()]  
G = [int(i) for i in G.split()]  
  
F = list(set(F))  
G = list(set(G))  
  
F.sort()  
G.sort()  
  
insereaza\_cifra\_control(F)  
insereaza\_cifra\_control(G)  
  
F1 = [F[i] for i in range(1, len(F), 2)]  
G1 = [G[i] for i in range(1, len(G), 2)]  
  
if egale(F1, G1) == True:  
 print("Da")  
else:  
 print("Nu")  
  
f.close()  
g.close()

4) Spiridușii lui Moș Crăciun au fiecare asociat câte un cod format din litere și cifre. Până în luna noiembrie spiridușii au avut la dispoziție un fișier în care să completeze ce jucării pot face până la Crăciun și câte bucăți. O linie din acest fișier conține codul spiridușului, numărul de bucăți (număr natural) și numele jucăriei (numele este format din cuvinte separate prin câte un spațiu). Un spiriduș poate adăuga de mai multe ori o linie în fișier, chiar și cu aceeași jucărie, dacă se hotărăște că poate face mai multe. Un exemplu de fișier este:

S1 1 papusa

S2 1 papusa

S3 1 masinuta

S1 10 trenulet

S2 1 papusa

S2 2 masinuta

S1 10 ponei

S3 15 ponei

a) Memorați datele din fișier astfel încât Moș Crăciun să poată afla cât mai repede informațiile cerute la punctele următoare.

f = open("numere.in")  
d={}  
for linie in f.readlines():  
 cod\_spiridus, cantitate, tip = linie.split()  
 if cod\_spiridus not in d:  
 d[cod\_spiridus] = {}  
 if tip not in d[cod\_spiridus]:  
 d[cod\_spiridus][tip] = int(cantitate)  
 else:  
 d[cod\_spiridus][tip] += int(cantitate)

b) Dat codul unui spiriduș, care sunt jucăriile care le poate face și ce cantitate din fiecare? Pentru aceasta scrieți o funcție despre\_spiridus cu 2 parametri: în primul parametru se transmite structura în care s-au memorat datele la punctul a) iar al doilea este codul unui spiriduș. Funcția returnează o lista cu elementele tupluri de 2 elemente – primul fiind numele jucăriei, iar al doilea cantitatea – ordonată descrescător după cantitate și, în caz de egalitate, crescător după nume. Apelați funcția pentru codul S1 și afișați lista returnata de funcție. Pentru datele din fișierul exemplu se va afișa [('ponei', 10), ('trenulet', 10), ('papusa', 1)].

def despre\_spiridus(d, cod\_spiridus):  
 def cmp(tuplu):  
 return (-tuplu[1], tuplu[0])  
 items = d[cod\_spiridus].items()  
 items = list(items)  
 items.sort(key=cmp)  
 return items

c) Care este mulțimea jucăriilor pe care spiridușii le pot produce? Pentru aceasta scrieți o funcție jucarii care primește ca parametru structura în care s-au memorat datele la punctul a) și returnează o mulțime cu numele jucăriilor care pot fi produse de spiriduși. Apelați funcția și afișați pe ecran elementele mulțimii returnate (pe o linie, separate prin virgula). Pentru datele din fișierul exemplu o posibilă ieșire este (jucăriile se pot afișa în orice ordine): ponei,trenulet,masinuta,papusa

def jucarii(d):  
 s = set()  
 for elem in d:  
 keys = d[elem].keys()  
 for key in keys:  
 s.add(key)  
 print(", ".join(s))

d) Care este lista spiridușilor harnici: ordonați descrescător după numărul de jucării diferite pe care le pot face și, în caz de egalitate, descrescător după cantitatea de jucării pe care o vor produce și, în caz de egalitate, crescător după cod? Pentru aceasta scrieți o funcție spiridusi care primește ca parametru structura în care s-au memorat datele și returnează o lista cu elementele tupluri de 3 elemente – primul fiind codul spiridușului, al doilea numărul de jucării diferite pe care le poate produce, iar al treilea numărul total de bucăți de jucării care le poate produce spiridușul – ordonată după criteriile cerute de Moș Crăciun (precizate anterior). Apelați funcția și afișați pe ecran elementele listei obținute, fiecare tuplu din listă fiind afișat pe o linie separată. Pentru datele din fișierul exemplu se va afișa

('S1', 3, 21)

('S3', 2, 16)

('S2', 2, 4)

def spiridusi(d):  
 def cmp(tuplu):  
 return (-tuplu[1], -tuplu[2], tuplu[0])  
 L = []  
 for elem in d:  
 tuplu = []  
 tuplu.append(elem)  
 x = len(list(d[elem]))  
 tuplu.append(x)  
 tuplu.append(sum(list(d[elem].values())))  
 L.append(tuple(tuplu))  
 L.sort(key=cmp)  
 return L

d) În caz că un spiriduș nu mai poate produce un tip de jucărie, să se actualizeze informațiile asociate spiridușului. Pentru aceasta scrieți o funcție actualizare care primește 3 parametri (în aceasta ordine): structura în care s-au memorat datele, codul spiridușului și numele jucăriei. Dacă spiridușul dat ca parametru produce cel puțin două tipuri de jucării, funcția va șterge din structura de date informațiile despre jucăria cu numele dat ca parametru atașate spiridușului cu codul dat ca parametru și va returna True. Altfel funcția va returna False . Să se apeleze funcția pentru spiridușul cu cod S1 și jucăria trenulet, apoi să se afișeze informațiile rămase despre S1 folosind funcția despre\_spiridus de la punctul b. Pentru datele din fișierul exemplu se va afișa

[('ponei', 10), ('papusa', 1)]

def actualizare(d, cod\_spiridus, tip):  
 if len(list(d[cod\_spiridus])) >= 2:  
 if tip in d[cod\_spiridus]:  
 d[cod\_spiridus].pop(tip)  
 return True  
 else:  
 return False  
 else:  
 return False  
  
actualizare(d, 'S1', 'trenulet')  
print(despre\_spiridus(d,'S1'))

PROBL 3 A+B

1. **def** citire\_date(fisier):
2. **with** open(fisier, 'rt') **as** f:
3. date = f.read().splitlines()
5. *# d['S1'] = [('ponei', 10)] # nu*
6. *# d['S1']['ponei'] = 10*
8. *# cheie = cod\_spiridus*
9. *# valoare = dictionar*
10. *#               cheie = tip\_jucarie*
11. *#               valoare = cantitate*
13. **from** collections **import** defaultdict
14. d = defaultdict(dict)
15. **for** linie **in** date:
16. cod\_spiridus, cantitate, tip = linie.split(' ')
17. **if** tip **in** d[cod\_spiridus]:
18. d[cod\_spiridus][tip] += int(cantitate)
19. **else**:
20. d[cod\_spiridus][tip] = int(cantitate)
22. **return** d
24. **def** main():
25. date = citire\_date('spiridusi.txt')
26. **print**(date)
28. **if** \_\_name\_\_ == '\_\_main\_\_':
29. main()

SAU

d = {}

for linie in date:

cod\_spiridus, cantitate, tip = linie.split(' ')

if cod\_spiridus not in d:

d[cod\_spiridus] = {}

if tip not in d[cod\_spiridus]:

d[cod\_spiridus][tip] = int(cantitate)

else:

d[cod\_spiridus][tip] += int(cantitate)