

Subiectul 1 – limbajul Python – 3 p.

a) Scrieți o funcție *litere* care primește un număr variabil de cuvinte formate doar din litere mici și returnează un dicționar care conține, pentru fiecare cuvânt primit ca parametru, un dicționar conținând frecvențele literelor distincte din cuvântul respectiv sub forma unei perechi *cuvânt: {litere: frecvențe}*. De exemplu, pentru apelul *litere("teste", "programare")* funcția trebuie să furnizeze dicționarul {'teste': {'e': 2, 't': 2, 's': 1}, 'programare': {'g': 1, 'a': 2, 'm': 1, 'e': 1, 'o': 1, 'p': 1, 'r': 3}}. **(1.5 p.)**

b) Înlocuiți punctele de suspensie din instrucțiunea *numere = [...]* cu o secvență de inițializare (*list comprehension*) astfel încât, după executarea sa, lista să conțină numerele naturale formate din exact două cifre care sunt divizibile cu 2, dar nu sunt divizibile cu 6. **(0.5 p.)**

c) Considerăm următoarea funcție recursivă:

```
def f(v, p, u):
    if u == p:
        return 0
    else:
        m = (p+u)//2
        n1 = f(v, p, m)
        n2 = f(v, m+1, u)
        n3 = 0
        for i in range(p, m+1):
            for j in range(m+1, u+1):
                if v[i] > v[j]:
                    n3 += 1
        return n1 + n2 + n3
```

Determinați complexitatea funcției apelată pentru o listă **L** formată din **n** numere întregi astfel: **f(L, 0, n-1)**. **(1 p.)**

Subiectul 2 – metoda Greedy (3 p.)

Complexitatea maximă a soluției: $\mathcal{O}(n \log_2 n)$

Gigel tocmai a învățat la școală adunarea și înmulțirea numerelor întregi. Pentru a-l ajuta pe Gigel să-și fixeze cunoștințele proaspăt dobândite, precum și pentru a-i testa istețimea algoritmică, bunicul său a scris pe mai multe cartonașe numere întregi nenule și apoi le-a împărțit în două grămezi: o grămadă A formată din m cartonașe și o grămadă B formată din n cartonașe ($1 \leq m \leq n \leq 100000$). Sarcina lui Gigel este să selecteze din grămada B exact m cartonașe pe care apoi să le împerecheze, în orice ordine dorește el, cu cele m cartonașe din grămada A astfel încât prin însumarea produselor celor două numerele scrise pe fiecare pereche de cartonașe să obțină cea mai mare sumă posibilă. Scrieți un program Python care citește de la tastatură valorile scrise pe cartonașele din cele două grămezi și afișează pe ecran cea mai mare sumă pe care o poate obține Gigel respectând restricțiile indicate în enunțul problemei, precum și o modalitate de obținere a sa în forma indicată în exemplu.

Exemplu:

Dacă $A = [3, -2, 5, -1, 4]$ și $B = [7, 8, -5, 2, -4, -1, 5]$, atunci suma maximă pe care o poate obține Gigel este $97 = 3 * 5 + (-2) * (-5) + 5 * 8 + (-1) * (-4) + 4 * 7$.

Subiectul 3 – metoda Programării Dinamice (3 p.)

Complexitatea maximă a soluției: $O(nm)$

O regiune deșertică poate fi reprezentată printr-un tablou cu m linii și n coloane. Elementele tabloului reprezintă diferențele de nivel față de nivelul mării măsurate în metri. Un beduin trebuie să traverseze deșertul de la un punct dat din sud (un element dat de pe ultima linie) la nord (orice element de pe prima linie). La fiecare pas el se poate deplasa într-unul dintre elementele vecine cu cel care se află, în una dintre direcțiile NE, N, NV. Un traseu este considerat optimal dacă numărul total de metri urcați de beduin este minim. Scrieți un program Python care citește de la tastatură indicele coloanei de unde pleacă beduinul (cu numerotare de la 0; amintim că beduinul pornește de pe ultima linie) și matricea care reprezintă deșertul, după care determină o modalitate optimală de a traversa deșertul. Se vor afișa numărul minim de metri urcați și un traseu optim, cu liniile și coloanele numerotate de la 0. În plus, determinați dacă traseul optim este unic și afișați un mesaj corespunzător.

Intrare de la tastatură	Ieșire pe ecran
1	1
1 3 4 2	3 1
0 2 3 2	2 1
2 1 1 1	1 0
1 1 3 1	0 0
	traseul optim nu este unic

Subiectul 4 – metoda Backtracking (3 p.)

a) După o lungă plimbare prin parc cu stăpâna sa, cățelușă Laika se află în fața unei mari provocări: trebuie să urce cele n trepte ($1 \leq n \leq 50$) până la ușa apartamentului în care locuiește. Din cauza oboselii, Laika poate să sară, de fiecare dată, peste un număr de trepte cuprins între 1 și t ($1 \leq t \leq n$). Fiind foarte curioasă, cățelușă Laika se gândește cum ar putea să afle toate modalitățile în care ar putea să urce cele n trepte. Ajutați-o pe Laika, scriindu-i un program Python care să citească de la tastatură numerele naturale n și t , după care să-i afișeze toate modalitățile corecte de urcare (evident, Laika știe să citească) și numărul acestora! **(2.5 p.)**

Exemplu:

Pentru $n = 5, t = 3$ există 13 modalități în care Laika poate urca treptele:

1, 1, 1, 1, 1

1, 1, 1, 2

1, 1, 2, 1

1, 1, 3

1, 2, 1, 1

1, 2, 2

1, 3, 1

2, 1, 1, 1

2, 1, 2

2, 2, 1

2, 3

3, 1, 1

3, 2

b) Precizați cum ar trebui modificată o singură instrucțiune din program astfel încât să afișeze doar soluțiile în care soluțiile în care Laika sare, de fiecare dată, peste un număr de trepte cel mult egal cu cel precedent. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. **(0.5 p.)**