

## MODALITATEA DE DESFĂȘURARE A EXAMENULUI LA DISCIPLINA "PROGRAMAREA ALGORITMILOR"

- Examenul la disciplina "Programarea algoritmilor" se va desfășura în ziua de 20.01.2022, între orele 9<sup>00</sup> și 11<sup>30</sup>, astfel:
  - 09<sup>00</sup> – 09<sup>30</sup>: efectuarea prezenței studenților
  - 09<sup>30</sup> – 11<sup>30</sup>: desfășurarea examenului
  - 11<sup>30</sup> – 12<sup>00</sup>: verificarea faptului că sursele trimise de către studenți au fost salvate pe platforma MS Teams
- Testul se va desfășura pe platforma MS Teams, iar pe tot parcursul desfășurării sale, de la ora 09<sup>00</sup> la ora 12<sup>00</sup>, studenții trebuie să fie conectați pe canalul dedicat cursului de "Programarea algoritmilor" corespunzător seriei lor.
- În momentul efectuării prezenței, fiecare student trebuie să aibă pornită camera video în MS Teams și să prezinte buletinul sau cartea de identitate. Dacă dorește să-și protejeze datele personale, studentul poate să acopere codul numeric personal și/sau adresa!
- În timpul desfășurării testului studenții pot să închidă camera video, dar trebuie să o deschidă dacă li se solicită acest lucru de către un cadru didactic!
- Toate subiectele se vor rezolva folosind limbajul Python.
- Subiectul 1 este obligatoriu, iar dintre subiectele 2, 3 și 4 se vor rezolva CEL MULT DOUĂ, la alegere.
- Citirea datelor de intrare se va realiza de la tastatură, iar rezultatele vor fi afișate pe ecran.
- Se garantează faptul că datele de intrare sunt corecte.
- Operațiile de sortare se vor efectua folosind funcții sau metode predefinite din limbajul Python.
- Pentru subiectul 1 nu contează complexitatea soluției propuse.
- Rezolvările subiectelor alese dintre subiectele 2, 3 și 4 trebuie să conțină:
  - o scurtă descriere a algoritmului și o argumentare a faptului că acesta se încadrează într-o anumită tehnică de programare;
  - în cazul problemelor rezolvate folosind metoda Greedy sau metoda programării dinamice se va argumenta corectitudinea criteriului de selecție sau a relațiilor de calcul;
  - în cazul subiectelor unde se precizează complexitatea maximă pe care trebuie să o aibă soluția, se va argumenta complexitatea soluției propuse și vor primi punctaj maxim doar soluțiile corecte care se încadrează în complexitatea cerută;
  - în cazul problemei rezolvate folosind metoda backtracking nu contează complexitatea soluției propuse, dar se va ține cont de eficiența condițiilor de continuare;
  - în fiecare program Python se va preciza, pe scurt, sub forma unor comentarii, semnificația variabilelor utilizate.
- Rezolvările corecte care nu respectă restricțiile indicate vor primi punctaje parțiale.
- Se acordă 1 punct din oficiu.
- Rezolvările tuturor subiectelor se vor scrie de mână, folosind pix/stilou cu culoarea pastei/cernelii albastră sau neagră. Pe fiecare pagina studentul își va scrie numele și grupa, iar paginile trebuie să fie numerotate.
- Înainte de expirarea timpului alocat examenului, toate paginile vor fi fotografiate/scanate clar, în ordinea corectă, și transformate într-un singur fișier PDF care va fi încărcat în Google Drive folosind un anumit formular.
- Numele fișierului PDF trebuie să respecte șablonul *grupa\_nume\_prenume.pdf*. De exemplu, un student cu numele Popescu Ion Mihai din grupa 131 trebuie să denumească fișierul care conține rezolvările tuturor subiectelor astfel: *131\_Popescu\_Ion\_Mihai.pdf*.

### Subiectul 1 – limbajul Python – 3 p.

a) Scrieți o funcție **aparitii** care primește un număr variabil de numere naturale și returnează un dicționar care conține pentru fiecare număr primit ca parametru o listă de perechi în care pentru fiecare cifră distinctă a numărului avem perechea formată din valoarea cifrei și frecvența sa în acel număr. De exemplu, pentru apelul **aparitii(1011, 234, 8158558)** funcția trebuie să returneze dicționarul {1011: [(1,3), (0,1)], 234: [(2,1), (3,1), (4,1)], 8158558: [(1,1), (5,3), (8,3)]}. **(1.5 p.)**

b) Considerând un dicționar cu același format ca dicționarul de la punctul a), scrieți o secvență de inițializare pentru o listă (*list comprehension*) care să conțină cheile din dicționar care au cel puțin 3 cifre distincte. De exemplu, pentru dicționarul anterior, se va obține lista [234, 8158558]. **(0.5 p.)**

c) Considerăm următoarea funcție recursivă:

```
def f(lista, p, u):  
    if u - p <= 2:  
        return sum(lista[p: u+1])  
    k = (u - p + 1) // 3  
    aux_1 = sum(lista[p: p + k])  
    aux_2 = f(lista, p + k + 1, p + 2 * k - 1)  
    aux_3 = sum(lista[p+2*k+1: u+1])  
    return sum([aux_1, aux_2, aux_3])
```

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:  $O(n \log_2 n)$

Bunicul vostru are  $N$  recipiente care conțin diverse cantități de apă la diferite temperaturi, respectiv recipientul  $i$  conține  $X_i$  litri de apă la temperatura  $Y_i$ . Bunicul vostru dorește să facă baie și crede cu tărie faptul că apa în care se va spăla trebuie să aibă exact o temperatură  $T$ . De asemenea, bunicul consideră că este cu atât mai bine cu cât are mai multă apă la dispoziție. Din păcate, pe bunic nu îl ține spatele, așa că vă roagă pe voi să alegeți inteligent ce cantitate de apă din fiecare recipient trebuie să turnați în cadă astfel încât bunicul vostru să se poată bucura de o baie plăcută! Temperatura apei nu este afectată de mediul exterior. La final, dacă ați turnat  $0 \leq C_i \leq X_i$  din recipiente, temperatura finală a apei din cadă va fi egală cu:

$$F = \frac{C_1 * Y_1 + C_2 * Y_2 + \dots + C_N * Y_N}{C_1 + C_2 + \dots + C_N}$$

Datele de intrare se vor citi de la tastatură, astfel:

- de pe prima linie două numere separate printr-un spațiu, respectiv numărul natural strict pozitiv  $N$  de recipiente și numărul real  $T$  reprezentând temperatura dorită de bunicul vostru pentru apa din cadă;
- de pe următoarele  $N$  linii se vor citi câte două numere reale  $X_i$  și  $Y_i$  cu semnificația din enunț (cantitatea de apă din recipientul  $i$  și temperatura sa).

Programul Python trebuie să afișeze pe ecran cele  $N$  valori reale  $C_i$ , fiecare pe câte o linie, cu exact 3 zecimale. Se acceptă orice soluție corectă.

**Exemplu:**

Date de intrare	Date de ieșire
3 40	50.000
50 20	72.500
72.5 50	5.500
38.72 90	

**Explicații:** În cadă s-au turnat exact  $50.000 + 72.5 + 5.5 = 128$  litri, iar temperatura finală a apei este  $(50.000 * 20 + 72.5 * 50 + 5.5 * 90) / 128 = 5120 / 128 = 40$  de grade. Nu există nicio altă variantă prin care să se toarne în cadă mai mult de 128 de litri și apa să aibă exact temperatura de 40 de grade!

### Subiectul 3 – metoda Programării Dinamice (3 p.)

Complexitatea maximă a soluției:  $O(n^2)$

Schiorel a urcat cu telecabina până în vârful stațiunii și își dorește să ajungă cât mai obosit la una din cabanele stațiunii ca să se poată hidrata cât mai intens. Stațiunea e reprezentată ca o matrice pătratică de dimensiune  $n$  în care în fiecare pătrat avem gradul de oboseală pe care îl va acumula Schiorel dacă trece prin acel câmp sau -1, însemnând că în acel câmp avem o cabană. Schiorel poate începe traseul de oriunde de pe linia de sus și se poate opri la orice cabană, voi trebuie să-l ajutați să ajungă cât mai obosit! Schiorel poate coborî drept sau în diagonală, adică din  $(i,j)$  în  $\{(i+1,j-1), (i+1, j), (i+1, j+1)\}$  evident fără a părăsi stațiunea.

Scrieți un program Python care citește de la tastatură dimensiunea tablei  $n$  și pentru fiecare pătrățică de coordonate  $(i,j)$  (cu  $i=1,...,n, j=1,...,n$ ) o valoare  $c_{ij}$  cu semnificația:

- dacă  $c_{ij}$  este număr natural, el reprezintă gradul de oboseală acumulat de Schiorel când trece prin acea zona a stațiunii.
- dacă  $c_{ij}$  este -1, atunci în acea zonă se află o cabană!

și afișează un traseu al lui Schiorel până la o cabană, astfel încât să ajungă cât mai obosit (odată ajuns la o cabană, Schiorel se oprește și nu mai continuă drumul).

Intrare de la tastatură	Ieșire pe ecran
4 5     2     6     11 -1    7     1     -1 4    10     3     5 1     6    -1     2	Gradul de oboseala maxim 23 1 3 2 2 3 2 4 3

*Explicații:* Părtia este o matrice de dimensiuni 4x4 în care elementele reprezintă oboseala acumulată trecând prin acel punct, respectiv -1 în locul în care avem cabană. Pe traseul (1,3), (2,2), (3,2), (4,3) acumulează oboseala 23 (!traseul trebuie să înceapă pe prima linie și să se termine cu o pătrățică de valoare -1).

Intrare de la tastatură	Ieșire pe ecran
4 5     2     6    31 -1    7    -1    -1 4    10     3     5 1     6    -1     2	Gradul de oboseala maxim 31 1 4 2 3

Odată ajuns la o cabana, Schiorel se oprește din schiat.

P.S. Lui Schiorel îi place oboseala.

#### Subiectul 4 – metoda Backtracking (3 p.)

a) Petrișor ar vrea să își schimbe parolele și are o mulțime de litere preferate **L** și o mulțime **S** de simboluri (caractere care nu sunt litere) pe care ar vrea să le folosească în parole pentru a crește siguranța acestora. Pentru a îi fi mai ușor să le țină evidența, el ar vrea să își construiască parole de aceeași lungime după un anumit tipar. Tiparul este un șir de caractere de lungime **n** format doar cu caracterele '**l**' și '**s**' cu semnificația: dacă în tipar pe poziția **i** este caracterul '**l**', atunci în parolă pe poziția **i** va fi o literă din mulțimea **L**, iar dacă în tipar pe poziția **i** este caracterul '**s**', atunci în parolă pe poziția **i** va fi un simbol din mulțimea **S**. Mai mult, Petrișor și-ar dori ca orice simbol din **S** și orice literă din **L** să apară cel mult o dată în parolă. Scrieți un program Python care să citească de la tastatură numărul **n**, tiparul **T** și mulțimile **L** și **S**, după afișează toate parolele care verifică cerințele lui Petrișor sau mesajul "*Imposibil*" dacă nu există nicio parolă având proprietățile cerute. (2.5 p.)

**Exemplu:** Pentru **n = 6**, tiparul '**lslsll**', mulțimea **L** de litere '**a**', '**b**', '**c**', '**D**' și mulțimea **S** de simboluri '@', '.' trebuie afișate următoarele 48 de parole (nu neapărat în această ordine):

a@b.cD	b@D.ca	c.D@ab
a@b.Dc	b.a@cD	c.D@ba
a@c.bD	b.a@Dc	D@a.bc
a@c.Db	b.c@aD	D@a.cb
a@D.bc	b.c@Da	D@b.ac
a@D.cb	b.D@ac	D@b.ca
a.b@cD	b.D@ca	D@c.ab
a.b@cD	c@a.bD	D@c.ba
a.c@bD	c@a.Db	D.a@bc
a.c@Db	c@b.aD	D.a@cb
a.D@bc	c@b.Da	D.b@ac
a.D@cb	c@D.ab	D.b@ca
b@a.cD	c@D.ba	D.c@ab
b@a.Dc	c.a@bD	D.c@ba
b@c.aD	c.a@Db	
b@c.Da	c.b@aD	
b@D.ac	c.b@Da	

b) Modificați o singură instrucțiune din program astfel încât să fie afișate doar parolele care încep cu o vocală. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. (0.5 p.)