

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 9⁰⁰ și 11³⁰, astfel:
 - 09⁰⁰ – 09³⁰: efectuarea prezenței studenților
 - 09³⁰ – 11³⁰: desfășurarea examenului
 - 11³⁰ – 12⁰⁰: 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 09⁰⁰ la ora 12⁰⁰, 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) Știind că matricea pătratică **m** este memorată sub forma unei liste de liste, înlocuiți punctele de suspensie din instrucțiunea **numere = [...]** cu o secvență de inițializare (*list comprehension*) astfel încât, după executarea sa, lista să conțină pătratul elementelor aflate pe diagonala principală a matricei **m**. De exemplu, pentru matricea **m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]** trebuie ca lista **numere** să fie [1, 25, 81]. **(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: $O(n^2)$

După o nouă plimbare lungă prin parc cu stăpâna sa, cățelușa Laika se află în fața unei mari provocări: trebuie iar să urce cele n trepte până la ușa apartamentului în care locuiește. De data aceasta ea mai are însă energie și poate să sară de pe o treaptă i direct pe una dintre treptele $i+1, i+2, \dots, n$. Totuși pentru a sari i trepte trebuie să plătească un cost c_i , pentru că face gălăgie. De asemenea, pentru că treptele au fost spălate și Laika vine după o joacă prin noroi, pentru fiecare treaptă i ($i=1, \dots, n$) pe care calcă Laika există un cost t_i (număr natural pozitiv) pe care trebuie să îl plătească. Ajutați-o pe Laika scriind un program Python care afișează o modalitate de a urca treptele de la baza scării (considerată treapta 0, pentru care taxa este $t_0 = 0$) la treapta n cu cost total minim. Se citesc de la tastatură n și taxele pentru cele n trepte t_1, t_2, \dots, t_n și costul pentru a sări treptele c_1, c_2, \dots, c_n (c_i este costul pe care trebuie să îl plătească dacă sare i trepte). Laika nu e un câine normal așa că s-ar putea costul să sară 3 trepte să fie mai mic decât să sară 2.

Intrare de la tastatură	Ieșire pe ecran
5 1 2 11 1 1 1 7 4 9 5	taxa totala 6 pentru traseul cu scările 0 5

Explicații: $n=5$, deci scara are 5 trepte numerotate 1,2, ... ,5 (pe lângă baza scării care se consideră treapta 0 și pentru care nu se plătește taxă); Laika preferă să sară din prima 5 trepte pentru că poate :), deci costul plătit va fi $c_5 + t_5 = 5 + 1 = 6$ (c_5 pentru că a sărit 5 trepte și t_5 pentru că a călcat pe treapta 5)

Intrare de la tastatură	Ieșire pe ecran
5 1 2 11 1 1 1 7 4 9 15	taxa totala 9 pentru traseul cu scările 0 1 4 5

Explicații: Laika sare la treapta 1 cu costul 2 ($c_1=1$ costul de a sări o treaptă, $t_1=1$ costul de a ajunge pe treapta 1), apoi Laika sare la treapta 4 cu costul 5 ($c_3=4$ costul de a sări 3 trepte, $t_4=1$ costul de călca pe treapta 4), apoi sare la treapta 5 cu costul 2 ($1+1$).

Subiectul 4 – metoda Backtracking (3 p.)

a) Un număr natural se numește *p-mărginit* ($0 \leq p \leq 9$) dacă valoarea absolută a diferenței dintre oricare două cifre ale sale este cel mult egală cu p . De exemplu, numărul 27383 este *6-mărginit*, iar numărul 2022 este *2-mărginit*. Scrieți un program Python care să citească de la tastatură numerele naturale p și c , după care afișează toate numerele naturale *p-mărginite* formate din cifre nenule având suma cifrelor egală cu c sau mesajul "Imposibil" dacă nu există niciun astfel de număr. **(2.5 p.)**

Exemplu:

Pentru $p = 3$ și $c = 6$ trebuie afișate următoarele 30 de numere (nu neapărat în această ordine):

111111	1221	222
11112	123	231
11121	1311	24
1113	132	3111
11211	141	312
1122	21111	321
1131	2112	33
114	2121	411
12111	213	42
1212	2211	6

b) Precizați cum ar trebui modificată o singură instrucțiune din program astfel încât să fie afișate doar numerele *p-mărginite* formate din cifre nenule având suma cifrelor egală cu c și prima cifră egală cu ultima. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. **(0.5 p.)**