

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 **cmmdc** care primește un număr variabil de tuple format din câte două numere naturale nenule și returnează pentru fiecare tuple primit ca parametru cel mai mare divizor comun al elementelor sale, sub forma unui dicționar cu perechi de tipul **cel mai mare divizor: lista tuplurilor care au cel mai mare divizor comun egal cu cheia**. De exemplu, pentru apelul **cmmdc((12, 50), (3, 4), (11, 13), (14, 2))** funcția trebuie să furnizeze dicționarul {2: [(12, 50), (14, 2)], 1: [(3, 4), (11, 13)]}. **(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 nu sunt pătrate perfecte și nici divizibile cu 7. **(0.5 p.)**

c) Considerăm următoarea funcție recursivă:

```
def f(lista):  
    if len(lista) <= 2:  
        return min(lista)  
    k = len(lista) // 2  
    aux_1 = lista[:k]  
    aux_2 = lista[k+1:]  
    return min(f(aux_1), f(aux_2), lista[k])
```

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

Subiectul 2 – metoda Greedy (3 p.)

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

O balanță veche s-a defectat și acum se echilibrează nu doar pentru două obiecte având aceeași greutate, ci pentru orice două obiecte cu proprietatea că modulul diferenței dintre greutatea lor este mai mic sau egal decât un număr real g . Scrieți un program Python care citește de la tastatură un număr natural n , un număr real g și greutatea a n obiecte, după care afișează pe ecran numărul maxim de perechi de obiecte care echilibrează balanța defectă, precum și perechile respective, știind că orice obiect poate să facă parte din cel mult o pereche. Fiecare pereche afișată trebuie să fie de forma $x + y$, unde x și y sunt numerele de ordine ale celor două obiecte din pereche (obiectele sunt numerotate începând de la 1). Greutățile tuturor obiectelor și diferența g sunt exprimate prin numere reale strict pozitive, reprezentând grame. Nu contează ordinea în care se vor afișa perechile de obiecte pe ecran și nici ordinea numerelor de ordine ale obiectelor dintr-o pereche.

Exemplu:

Date de intrare	Date de ieșire
10	3
8.5	3 + 2
21.25	10 + 4
12	1 + 8
6.05	
20.7	
23.8	
22	
33.25	
21	
48.15	
62.20	

Explicații: Avem $n = 10$ și $g = 8.5$. Se pot forma maxim 3 perechi de obiecte care pot echilibra balanța defectă. Soluția nu este unică, o altă soluție corectă obținându-se, de exemplu, înlocuind perechea 1 + 6 cu perechea 1 + 5.

Subiectul 3 – metoda Programării Dinamice (3 p.)
Complexitatea maximă a soluției: $O(mn)$

Greierașul și-a propus să fie harnic vara aceasta și să adune singur grăunțe pentru iarnă, să nu mai ceară de la furnici. El pornește să adune grăunțe pe un câmp de forma unei table dreptunghiulare cu m linii și n coloane, formată din pătrățele în care se pot găsi grăunțe sau furnici. Din fiecare pătrățică cu grăunțe pe care ajunge Greierașul ia toate grăunțele. Sunt însă și pătrățele în care Greierașul se întâlnește cu furnicile și acesta îi cer să le dea înapoi grăunțele cu care l-au împrumutat vara trecută. El poate trece de o pătrățică în care se află furnicuțe doar dacă are suficiente grăunțe să le dea. Greierașul pornește din colțul din dreapta sus al pădurii și se poate deplasa doar la vest, sud sau sud-vest, în una dintre pătrățele vecine cu cea în care se află. Greierașul pornește din colțul din dreapta sus și **se poate opri în orice pătrățică de pe ultima linie** să își construiască adăpost pentru iarnă unde să își depoziteze grăunțele adunate. Scrieți un program Python care citește de la tastatură dimensiunile tablei m și n și pentru fiecare pătrățică de coordonate (i,j) (cu $i=1,\dots,m$, $j=1,\dots,n$) o valoare c_{ij} cu semnificația:

- dacă $c_{ij} \geq 0$, atunci c_{ij} este numărul de grăunțe din pătrățică
- dacă $c_{ij} < 0$, atunci în pătrățica (i,j) se află o furnicuțe care îi cer $-c_{ij}$ grăunțe,

și afișează un traseu al Greierașului pe câmp astfel încât să adune un număr maxim de grăunțe (sub forma indicată în exemplu) .

Intrare de la tastatură	Ieșire pe ecran
4 4 5 2 3 1 -14 7 1 -2 1 -10 -3 15 -1 6 12 2	maxim 20 graunte pe traseul 1 4 1 3 2 3 3 3 4 3 4 2

Explicații: Pădurea este o matrice de dimensiuni 4×4 în care elementele pozitive sunt grăunțe care se pot aduna, iar cele negative sunt celule cu furnicuțe care cer un număr de grăunțe egal cu modulul numărului înscris în pătrățică. Traseul $(1,4)$, $(2,4)$, $(3,4)$, $(4,4)$, $(4,3)$, $(4,2)$, deși are suma $1 + (-2) + 15 + 2 + 12 + 6 = 34$, nu este valid deoarece Greierașul are o grăunță când ajunge în celula $(2,4)$, iar furnicuțele cer două.

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.)**