

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 **pareimpare** care primește un număr variabil de liste formate doar din numere naturale și returnează un dicționar care conține, pentru fiecare listă primită ca parametru, o pereche de forma **indicele listei: (listă cu numere impare, listă cu numere pare)**. Indicele listei reprezintă poziția la care apare printre parametri (pe poziția 0 apare prima listă primită ca parametru, pe poziția 1 apare a doua listă primită ca parametru etc.). În cazul în care nu există numere pare sau impare în listă, se adaugă lista vidă. De exemplu, pentru apelul **pareimpare([1, 1, 2, 3, 4], [0, 2], [1, 2, 3])** funcția trebuie să furnizeze dicționarul {0: ([1, 1, 3], [2, 4]), 1: ([], [0, 2]), 2: ([1, 3], [2])}. **(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 trei cifre care nu sunt divizibile cu 5 și sunt palindromuri. **(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: $O(n^2)$

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 pătrătică 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 pătrătică 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) Un număr natural se numește *echilibrat* dacă valoarea absolută a diferenței dintre oricare două cifre aflate pe poziții consecutive și, respectiv, între prima și ultima sa cifră este 0 sau 1. De exemplu, numerele 5456 și 1232 sunt echilibrate, iar numerele 5443, 5446 și 1233 nu sunt echilibrate. Scrieți un program Python care citește de la tastatură un număr natural m ($2 \leq m \leq 20$) și afișează toate numerele naturale *echilibrate* formate din exact m cifre, precum și numărul acestora. **(2.5 p.)**

Exemplu:

Pentru $m = 4$ trebuie afișate următoarele numere (nu neapărat în această ordine):

1000	3232	5456	7767
1001	3233	5544	7776
1010	3234	5545	7777
1011	3322	5554	7778
1012	3323	5555	7787
1100	3332	5556	7788
1101	3333	5565	7876
1110	3334	5566	7877
1111	3343	5654	7878
1112	3344	5655	7887
1121	3432	5656	7888
1122	3433	5665	7898
1210	3434	5666	8767
1211	3443	5676	8777
1212	3444	6545	8778
1221	3454	6555	8787
1222	4323	6556	8788
1232	4333	6565	8789
2101	4334	6566	8877
2111	4343	6567	8878
2112	4344	6655	8887
2121	4345	6656	8888
2122	4433	6665	8889
2123	4434	6666	8898
2211	4443	6667	8899
2212	4444	6676	8987
2221	4445	6677	8988
2222	4454	6765	8989
2223	4455	6766	8998
2232	4543	6767	8999
2233	4544	6776	9878
2321	4545	6777	9888
2322	4554	6787	9889
2323	4555	7656	9898
2332	4565	7666	9899
2333	5434	7667	9988
2343	5444	7676	9989
3212	5445	7677	9998
3222	5454	7678	9999
3223	5455	7766	Nr. solutii:159

b) Precizați cum ar trebui modificată o singură instrucțiune din program astfel încât să fie afișate doar numerele *echilibrate* formate din m cifre și care sunt numere pare. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. **(0.5 p.)**