

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 **litere** care primește un număr variabil de cuvinte formate din litere mici ale alfabetului englez și returnează un dicționar care conține pentru fiecare cuvânt primit ca parametru un dicționar cu frecvența fiecărei litere distincte care apare în cuvânt. De exemplu, pentru apelul **litere ('teste', 'dicționar', 'ele')**, funcția trebuie să returneze dicționarul {'teste': {'e': 2, 's': 1, 't': 2}, 'dicționar': {'a': 1, 'c': 1, 'd': 1, 'i': 2, 'n': 1, 'o': 1, 'r': 1, 't': 1}, 'ele': {'e': 2, 'l': 1}}. **(1.5 p.)**

b) Înlocuiți punctele de suspensie din instrucțiunea `lung = [...]` cu o secvență de inițializare (*list comprehension*) astfel încât, după executarea sa, lista să conțină toate tuplurile de forma (cuvânt, lungime) pentru fiecare cuvânt care începe cu o vocală dintr-o propoziție dată **p**. Propoziția este formată doar din litere mici ale alfabetului englez, iar cuvintele care o formează sunt distincte și despărțite între ele prin câte un spațiu. De exemplu, pentru propoziția `p = 'un exemplu de propozitie'`, lista rezultată va fi `lung = [('un', 2), ('exemplu', 7)]`. **(0.5 p.)**

c) Considerăm următoarea funcție recursivă:

```
def f(lista, p, u):
    if u - p <= 1:
        return lista[0]
    k = (p + u) // 2
    if k % 2 == 1:
        return f(lista, p, k-1) + f(lista, k+1, u)
    else:
        return f(lista, p, k-2) + f(lista, k+1, u)
```

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) 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.)**