

MODALITATEA DE DESFĂȘURARE A EXAMENULUI LA DISCIPLINA "PROGRAMAREA ALGORITMILOR"

- Examenul la disciplina "Programarea algoritmilor" se va desfășura în ziua de 19.01.2021, între orele 9³⁰ și 12⁰⁰, astfel:
 - 09³⁰ – 10⁰⁰: efectuarea prezenței studenților
 - 10⁰⁰ – 12⁰⁰: desfășurarea examenului
 - 12⁰⁰ – 12³⁰: 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 09³⁰ la ora 12³⁰, 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.
- 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 fiecare program Python se va preciza, pe scurt, sub forma unor comentarii, semnificația variabilelor utilizate.
- Pentru subiectele 1 nu contează complexitățile soluțiilor propuse.
- 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 pasteii/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 pe platforma MS Teams folosind un anumit formular.
- Numele fișierului PDF trebuie să respecte șablonul *grupa_nume_prenume_subiect.pdf*. De exemplu, un student cu numele Popescu Ion Mihai din grupa 131 trebuie să denumească fișierul care conține rezolvarea primului subiect astfel: *131_Popescu_Ion_Mihai_1.pdf*.

Subiectul 1 – limbajul Python – 3 p.

a) Scrieți o funcție *frecvențe* care primește un număr variabil de liste formate din numere întregi și returnează un dicționar cu perechi de forma *frecvență: lista valori* conținând toate frecvențele distincte ale tuturor elementelor tuturor listelor, precum și câte o listă cu elementele care au frecvența respectivă. De exemplu, pentru apelul *frecvente*([20, 10, 40, 20], [10, 20, 10], [40, 30, 40]) funcția trebuie să furnizeze dicționarul {3: [20, 10, 40], 1: [30]}. **(1.5 p.)**

b) Înlocuiți punctele de suspensie din instrucțiunea *tupluri* = [...] cu o secvență de inițializare (*list comprehension*) astfel încât, după executarea sa, lista să conțină pentru fiecare număr cuprins între 1 și 20 cu proprietatea că restul împărțirii sale la 6 este diferit de câtul împărțirii sale la 6 câte un tuplu format din numărul respectiv, câtul și restul împărțirii sale la 6. **(0.5 p.)**

c) Considerăm următoarea funcție recursivă:

```
def f(v, p, u):
    if u == p:
        return v[u]
    else:
        m = (p+u)//2
        s1 = f(v, p, m)
        s2 = 0
        for i in range(m+1, u+1):
            s2 = s2 + v[i]
        return s1 + s2
```

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: $\mathcal{O}(n \log_2 n)$

O tablă de joc are de-a lungul ei, pe mijloc, un șanț cu poziții numerotate cu numere consecutive, începând de la 1 la 10000. Pe tablă sunt plasate, în șanț, bețișoare cu capătul stâng într-o anumită poziție dată din șanț. Bețișoarele au lungimi date. Un jucător trebuie să aleagă bețișoarele pe care trebuie să le elimine pentru a rămâne pe tablă doar bețișoare care nu se ating între ele. Se consideră că două bețișoare care ocupă spațiul din șanț între pozițiile p_1 și p_2 , respectiv u_1 și u_2 se ating dacă intervalele $[p_1, p_2]$ și $[u_1, u_2]$ se intersectează. Un jucător câștigă dacă propune un număr minim de astfel de bețișoare care trebuie eliminate. Fiind date numărul n de bețișoare ($1 \leq n \leq 100000$) și pentru fiecare bețișor poziția capătului din stânga și lungimea lui, scrieți un program Python care să determine o mulțime de bețișoare care trebuie eliminate pentru a câștiga jocul (adică pe tablă să rămână un număr maxim de bețișoare care nu se ating între ele). Programul va afișa indicii bețișoarelor care trebuie eliminate, considerate ca fiind numerotate de la 1.

Exemplu:

Intrare de la tastatură	Ieșire pe ecran
5	1 3
3 70	
1 9	
1 19	
11 20	
40 40	

Explicații:

După eliminarea bețișoarelor 1 și 3 rămân: bețișorul 2 (care ocupă pozițiile de la 1 la 10), bețișorul 4 (care ocupă pozițiile de la 11 la 31) și bețișorul 5 (care ocupă pozițiile de la 40 la 80). Cele 3 bețișoare rămase nu se ating și nu există 4 bețișoare care să nu se atingă!

Subiectul 3 – metoda Programării Dinamice (3 p.)

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

Martinel a terminat sesiunea de examene și și-a propus ca mâine să se uite cât mai mult la televizor. Pentru aceasta și-a făcut o listă cu n emisiuni pe care ar vrea să le vadă. Pentru fiecare emisiune și-a notat intervalul de desfășurare $[s, t)$ și numele postului care difuzează emisiunea. Martinel vrea să vadă o emisiune de la început până la final (fără întrerupere, deci fără a schimba postul în timpul emisiunii). Scrieți un program Python care să îl ajute pe Martinel să aleagă o listă de emisiuni la care să se uite mâine cu suma duratelor maximă. Pentru aceasta, se citesc de la tastatură emisiunile de pe lista lui Martinel sub următoarea formă: pe câte o linie se dau informațiile despre câte o emisiune (separate între ele prin câte un spațiu), respectiv ora și minutul la care începe emisiunea, ora și minutul la care se termină, numele postului (poate conține spații). Programul va afișa o listă cu emisiunile selectate (având suma duratelor maximă), câte una pe linie, sub forma indicată în exemplu. În plus, determinați dacă soluția optimă este unică și afișați un mesaj corespunzător.

Intrare de la tastatură	Ieșire pe ecran
10 10 10 30 post 1	[9:10, 10:00) post 1
10 05 11 00 post 2	[10:10, 10:30) post 1
9 10 10 00 post 1	[10:40, 12:00) post 3
10 40 12 00 post 3	solutia este unica
11 40 12 10 post 2	

Subiectul 4 – metoda Backtracking (3 p.)

a) Temutul hacker Pufoshell a observat faptul că tot mai mulți utilizatori ai serviciului de email Hapciu! folosesc ca parole numere naturale formate din exact n cifre ($1 \leq n \leq 20$) așezate în ordine crescătoare și a căror sumă este egală cu o valoare s , un număr natural cuprins între 1 și 180, având, de obicei, o semnificație deosebită pentru întreaga omenire. Scrieți un program care să-i ajute pe utilizatorii serviciului de email Hapciu! să-și aleagă parole puternice, scriind un program Python care citește de la tastatură cele două numere naturale n și s , după care afișează parolele pe care le poate sparge Pufoshell sau un mesaj corespunzător dacă nu există nicio astfel de parolă. **(2.5 p.)**

Exemplu:

Pentru $n = 4$ și $s = 25$, toate parolele pe care le poate sparge Pufoshell sunt:

1888
2788
3688
3778
4588
4678
4777
5578
5668
5677
6667

b) Precizați unde ar trebui inserată în program o singură instrucțiune astfel încât să afișeze doar parolele care conțin cifra norocoasă 7. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. **(0.5 p.)**