

## 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 9<sup>30</sup> și 12<sup>00</sup>, astfel:
  - 09<sup>30</sup> – 10<sup>00</sup>: efectuarea prezenței studenților
  - 10<sup>00</sup> – 12<sup>00</sup>: desfășurarea examenului
  - 12<sup>00</sup> – 12<sup>30</sup>: 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 09<sup>30</sup> la ora 12<sup>30</sup>, 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 *ștergere* care primește o valoare  $x$  și un număr variabil de liste formate din numere întregi și returnează o listă care conține, pentru fiecare listă primită ca parametru, câte un tuplu de forma (*numărul aparițiilor lui  $x$  în listă, lista rămasă după ștergerea tuturor aparițiilor lui  $x$* ). De exemplu, pentru apelul *ștergere*(4, [2, 1, 4, 4], [-7, 3], [4, 4, 4]) funcția trebuie să furnizeze lista [(2, [2, 1]), (0, [-7, 3]), (3, [])]. **(1.5 p.)**

**b)** Înlocuiți punctele de suspensie din instrucțiunea *perechi* = [...] cu o secvență de inițializare (*list comprehension*) astfel încât, după executarea sa, lista să conțină toate tuplurile de forma ( $a, b$ ) cu proprietatea că  $a$  și  $b$  sunt cifre nenule cu parități diferite. **(0.5 p.)**

**c)** Considerăm următoarea funcție recursivă:

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

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:  $O(n^2)$

Pentru două cuvinte  $x$  și  $y$  formate doar din litere mici spunem că  $x$  este în relație cu  $y$  dacă ultima literă din  $x$  este alăturată (consecutivă) în alfabet cu prima literă din  $y$  (litera 'a' este alăturată literei 'b', litera 'b' este alăturată literelor 'a' și 'c' etc). Scrieți un program Python care să rezolve următoarea cerință:

Se citesc de la tastatură un număr natural  $n$  și un șir de  $n$  cuvinte formate doar cu litere mici ale alfabetului englez, date pe o singură linie și separate între ele prin câte un spațiu. Să se elimine din șir un număr minim de cuvinte astfel încât șirul rămas să verifice următoarea proprietate: orice cuvânt din șir, cu excepția ultimului, este în relație cu următorul cuvânt din șir. Se vor afișa pe ecran cuvintele eliminate. În plus, determinați dacă soluția optimă este unică și afișați un mesaj corespunzător.

Intrare de la tastatură	Ieșire pe ecran
7 casa apa bun bine fila dop orar	casa bun fila solutia nu este unica

*Explicații:* șirul rămas "apa", "bine", "dop", "orar" verifică proprietatea cerută.

Soluția nu este unică, o altă soluție ar fi fost, de exemplu: apa bun fila (atunci ar fi rămas șirul "casa", "bine", "dop", "orar")

#### 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.)**