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^{30} și 12^{00}, astfel:
  - 09^{30} – 10^{00}: efectuarea prezenței studenților
  - 10^{00} – 12^{00}: desfășurarea examenului
  - 12^{00} – 12^{30}: 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^{30} la ora 12^{30}, 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 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 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) Scrieţi o funcţie *apartine* care primeşte o mulţime (*set*) cu elemente numere întregi şi un număr variabil de liste formate din numere întregi şi returnează un dicţionar cu perechi de forma *număr din mulţime*: *lista de tupluri* (*indici, frecvenţă*) conţinând pentru fiecare număr din mulţime o listă de tupluri de forma (*indice, frecvenţă*) reprezentând indicii listelor primite ca parametru care îl conţin (prima listă are indicele 0) şi frecvenţa cu care apare în fiecare dintre aceste liste. De exemplu, pentru apelul *apartine* ({10,20}, [10, 11, 10], [20, 20, 40], [5], [10, 11]) funcţia trebuie să furnizeze dicţionarul {10: [(0, 2), (3, 1)], 20: [(1, 2)]} deoarece elementul 10 apare în lista 0 de două ori şi în lista 3 o dată, iar elementul 20 apare doar în lista 1 de două ori **(1.5 p.)**

b) Înlocuiţi punctele de suspensie din instrucţiunea *perechi* = […] cu o secvenţă de iniţializare (*list comprehension*) astfel încât, după executarea sa, lista să conţină toate tuplurile de forma (*a*, *b*) cu proprietatea că $0 < a < b < 11$. **(0.5 p.)**

c) Considerăm următoarea funcţie recursivă:

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

Determinaţi complexitatea funcţiei 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 beţisor 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	**Ieşire pe ecran**
5 3 70 1 9 1 19 11 20 40 40	1 3

Explicaţii:
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 sa vadă o emisiune de la început până la final (fără întrerupere, deci fără a schimba postul în timpul emisiunii). Scrieți un program Python care sa 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 10 05 11 00 post 2 9 10 10 00 post 1 10 40 12 00 post 3 11 40 12 10 post 2	[9:10, 10:00) post 1 [10:10, 10:30) post 1 [10:40, 12:00) post 3 solutia este unica

Subiectul 4 – metoda Backtracking (3 p.)

a) Gospodina Ana are d borcane cu dulceață ($1 \leq d \leq 20$), identificate prin numerele naturale de la 1 la d) și m borcane cu murături ($2 \leq m \leq 20$), identificate prin numerele de la d+1 la d+m. Scrieți un program Python care să citească de la tastatură cele două numere naturale d și m, după care să o ajute pe Ana să găsească toate modalitățile în care ar putea să așeze toate cele d+m borcane pe un raft, pe un singur rând, astfel încât să nu pună două borcane de murături unul lângă altul. Programul trebuie să afișeze un mesaj corespunzător dacă nu există nicio modalitate corectă de așezare a borcanelor pe raft. **(2.5 p.)**

Exemplu:
Pentru $d = 3$ și $m = 2$, borcanele se pot așeza corect astfel:

1, 2, 4, 3, 5	2, 4, 3, 1, 5	4, 1, 2, 3, 5	5, 1, 2, 3, 4
1, 2, 5, 3, 4	2, 4, 3, 5, 1	4, 1, 2, 5, 3	5, 1, 2, 4, 3
1, 3, 4, 2, 5	2, 5, 1, 3, 4	4, 1, 3, 2, 5	5, 1, 3, 2, 4
1, 3, 5, 2, 4	2, 5, 1, 4, 3	4, 1, 3, 5, 2	5, 1, 3, 4, 2
1, 4, 2, 3, 5	2, 5, 3, 1, 4	4, 1, 5, 2, 3	5, 1, 4, 2, 3
1, 4, 2, 5, 3	2, 5, 3, 4, 1	4, 1, 5, 3, 2	5, 1, 4, 3, 2
1, 4, 3, 2, 5	3, 1, 4, 2, 5	4, 2, 1, 3, 5	5, 2, 1, 3, 4
1, 4, 3, 5, 2	3, 1, 5, 2, 4	4, 2, 1, 5, 3	5, 2, 1, 4, 3
1, 5, 2, 3, 4	3, 2, 4, 1, 5	4, 2, 3, 1, 5	5, 2, 3, 1, 4
1, 5, 2, 4, 3	3, 2, 5, 1, 4	4, 2, 3, 5, 1	5, 2, 3, 4, 1
1, 5, 3, 2, 4	3, 4, 1, 2, 5	4, 2, 5, 1, 3	5, 2, 4, 1, 3
1, 5, 3, 4, 2	3, 4, 1, 5, 2	4, 2, 5, 3, 1	5, 2, 4, 3, 1
2, 1, 4, 3, 5	3, 4, 2, 1, 5	4, 3, 1, 2, 5	5, 3, 1, 2, 4
2, 1, 5, 3, 4	3, 4, 2, 5, 1	4, 3, 1, 5, 2	5, 3, 1, 4, 2
2, 3, 4, 1, 5	3, 5, 1, 2, 4	4, 3, 2, 1, 5	5, 3, 2, 1, 4
2, 3, 5, 1, 4	3, 5, 1, 4, 2	4, 3, 2, 5, 1	5, 3, 2, 4, 1
2, 4, 1, 3, 5	3, 5, 2, 1, 4	4, 3, 5, 1, 2	5, 3, 4, 1, 2
2, 4, 1, 5, 3	3, 5, 2, 4, 1	4, 3, 5, 2, 1	5, 3, 4, 2, 1

b) Precizați unde ar trebui inserată în program o singură instrucțiune astfel încât să afișeze doar modalitățile de așezare ale borcanelor pe raft care încep și se termină cu un borcan de dulceață, precum și instrucțiunea respectivă. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. **(0.5 p.)**