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 9^{00} și 11^{30}, astfel:
  - 09^{00} – 09^{30}: efectuarea prezenței studenților
  - 09^{30} – 11^{30}: desfășurarea examenului
  - 11^{30} – 12^{00}: 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^{00} la ora 12^{00}, 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) Scrieţi o funcţie **`litere`** care primeşte un număr variabil de cuvinte formate din litere mici ale alfabetului englez şi returnează un dicţionar care conţine pentru fiecare cuvânt primit ca parametru, un dicţionar cu frecvenţa fiecărei litere distincte care apare în cuvânt. De exemplu, pentru apelul **`litere('teste', 'dictionar', 'ele')`** funcţia trebuie să returneze dicţionarul `{'teste': {'e': 2, 's': 1, 't': 2}, 'dictionar': {'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) Folosind un dicţionar cu acelaşi format ca valorile dicţionarului de la punctul a) (i.e., cheile sunt litere, iar valorile frecvenţa literei respective), să se scrie o secvenţă de iniţializare (*list comprehension*) pentru o listă astfel încât aceasta să conţină perechile de forma **`(literă, frecvență)`** cu literele extrase din dicţionar care au frecvenţa pară. De exemplu, pentru dicţionarul `{'e': 2, 's': 1, 't': 2}` lista trebuie să fie `[('e', 2), ('t', 2)]`. **(0.5 p.)**

c) Considerăm următoarea funcţie recursivă:

```
def f(lista, p, u):
    if u-p <= 1:
        return sum(lista[p: u+1])
    k = (u-p+1) // 3
    aux_1 = f(lista, p, p+k)
    aux_2 = f(lista, p+k+1, p+2*k)
    aux_3 = f(lista, p+2*k+1, u)
    return aux_1 + aux_2 + aux_3
```

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: O(n $\log_2$ n)

Se organizează un concurs de tip trivia în limbajul Python la care participă nea Vasile împreună cu alți N oameni. Fiecărui om din cei N i se asociază o valoare strict pozitivă X_i, mai puțin lui Vasile căruia i se asociază implicit valoarea 0 (practic, Vasile este complet subestimat atât de către organizatori, cât și de ceilalți participanți). Concursul se desfășoară în mai multe runde eliminatorii, până când rămâne un singur om care este declarat câștigătorul. Într-o rundă la care participă K oameni (inclusiv Vasile!) sunt eliminați toți cei care nu știu răspunsul corect la întrebarea respectivă, iar scorul rundei se calculează ca fiind $\frac{\sum X_i}{K}$ pentru toți indicii i ai oamenilor eliminați în runda respectivă. Câștigătorul concursului va primi o sumă de bani egală cu suma scorurilor tuturor rundelor. Nea Vasile știe tot ceea ce s-ar putea ști despre limbajul Python (adică este imposibil ca el să piardă concursul!), deci îl interesează doar să afle suma maximă de bani pe care ar putea să o câștige.

Scrieți un program Python care să citească de la tastatură numărul natural nenul N, reprezentând numărul de concurenți (în afară de nea Vasile) și cele N numere strict pozitive asociate celor N participanți (separate între ele prin câte un spațiu), după care afișează un singur număr real (cu 3 zecimale) reprezentând suma maximă de bani pe care ar putea să o câștige nea Vasile.

Exemplu:

Date de intrare	Date de ieșire
2 6 6	5

Explicații: Suma maximă de bani pe care o poate câștiga Vasile este 5 = 2 + 3. În prima rundă ar trebui să iasă un singur adversar de-ai lui nea Vasile, scorul rundei fiind 6 / 3 = 2. În a doua rundă va ieși și ultimul adversar, scorul rundei fiind 6 / 2 = 3.

Subiectul 3 – metoda Programării Dinamice (3 p.)
Complexitatea maximă a soluției: O(nM)

Pia a hotărât să aloce *M* minute în care să facă doar activitățile ei preferate. Și-a făcut o listă cu *n* activități pe care le-a numerotat *1*,...,*n* și a estimat pentru fiecare activitate *i* =1,...,n durata d_i în minute. Ea ar vrea să își ocupe cât mai mult timp cu activitățile preferate și ar vrea să aleagă ce activități va face astfel încât timpul total dedicat activităților alese (egal cu suma duratelor lor) să fie cât mai apropiat de *M* (poate și depăși M, dar diferența între suma duratelor activităților alese și *M* trebuie să fie cât mai mică). Scrieți un program Python care să citească de la tastatură numărul de minute *M*, numărul de activități *n* și duratele d_1, ... , d_n și afișează ce activități să facă Pia astfel încât durata totală a acestora să fie cât mai apropiată de M.

Intrare de la tastatură	Ieșire pe ecran
21 6 5 10 5 4 10 3	1 3 5

Explicații: suma duratelor activităților 1, 3 și 5 este 5+5+10 = 20 și nu există o mulțime de activități cu suma duratelor 21. Soluția optimă nu este unică, o altă soluție optimă este de exemplu cea formată cu activitățile 2, 5 sau 3, 4, 5, 6 (în acest ultim caz durata totală este 22, cu 1 mai mare decât M, deci la fel de apropiată de M ca și 20)

Intrare de la tastatură	Ieșire pe ecran
20 4 10 11 4 12	1 2

Explicații: suma duratelor activităților 1, 2 este 21 și nu există o mulțime de activități cu suma duratelor 20 (deci cea mai apropiată durată totală de M=20 pe care o putem obține este 21)

Subiectul 4 – metoda Backtracking (3 p.)

a) Un număr natural se numeşte *p-mărginit* ($0 \leq p \leq 9$) dacă valoarea absolută a diferenţei 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*. Scrieţi un program Python care să citească de la tastatură numerele naturale *p* şi *c*, după care afişează 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 $p = 3$ şi $c = 6$ trebuie afişate 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) Precizaţi cum ar trebui modificată o singură instrucţiune din program astfel încât să fie afişate doar numerele *p-mărginite* formate din cifre nenule având suma cifrelor egală cu *c* şi prima cifră egală cu ultima. Pentru exemplul anterior, aceste soluţii sunt cele scrise cu roşu. **(0.5 p.)**