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) Înlocuiţi punctele de suspensie din instrucţiunea `lung = [...]` cu o secvenţă de iniţializare (*list comprehension*) astfel încât, după executarea sa, lista să conţină toate tuplurile de forma `(cuvânt, lungime)` pentru fiecare cuvânt care începe cu o vocală dintr-o propoziţie dată **p**. Propoziţia este formată doar din litere mici ale alfabetului englez, iar cuvintele care o formează sunt distincte şi despărţite între ele prin câte un spaţiu. De exemplu, pentru propoziţia `p ='un exemplu de propozitie'`, lista rezultată va fi `lung = [('un', 2), ('exemplu', 7)]`. **(0.5 p.)**

c) Considerăm următoarea funcţie 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)
```

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(mn)

Greierașul și-a propus să fie harnic vara aceasta și să adune singur grăunțe pentru iarnă, să nu mai ceară de la furnici. El pornește să adune grăunțe pe un câmp de forma unei table dreptunghiulare cu m linii și n coloane, formată din pătrățele în care se pot găsi grăunțe sau furnici. Din fiecare pătrățică cu grăunțe pe care ajunge Greierașul ia toate grăunțele. Sunt însă și pătrățele în care Greierașul se întâlnește cu furnicile și acesta îi cer să le dea înapoi grăunțele cu care l-au împrumutat vara trecută. El poate trece de o pătrățică în care se află furnicuțe doar dacă are suficiente grăunțe să le dea. Greierașul pornește din colțul din dreapta sus al pădurii și se poate deplasa doar la vest, sud sau sud-vest, în una dintre pătrățele vecine cu cea în care se află. Greierașul pornește din colțul din dreapta sus și **se poate opri în orice pătrățică de pe ultima linie** să își construiască adăpost pentru iarnă unde să își depoziteze grăunțele adunate. Scrieți un program Python care citește de la tastatură dimensiunile tablei m și n și pentru fiecare pătrățică de coordonate (i,j) (cu i=1,..,m, j=1,...,n) o valoare c_{ij} cu semnificația:

- dacă c_{ij} >=0, atunci c_{ij} este numărul de grăunțe din pătrățică
- daca c_{ij}<0, atunci în pătrățica (i,j) se află o furnicuțe care îi cer -c_{ij} grăunțe,

și afișează un traseu al Greierașului pe câmp astfel încât să adune un număr maxim de grăunțe (sub forma indicată în exemplu) .

Intrare de la tastatură	Ieșire pe ecran
4 4 5 2 3 1 -14 7 1 -2 1 -10 -3 15 -1 6 12 2	maxim 20 graunte pe traseul 1 4 1 3 2 3 3 3 4 3 4 2

Explicații: Pădurea este o matrice de dimensiuni 4x4 în care elementele pozitive sunt grăunțe care se pot aduna, iar cele negative sunt celule cu furnicuțe care cer un număr de grăunțe egal cu modulul numărului înscris în pătrățică. Traseul (1,4), (2,4), (3,4), (4,4), (4,3), (4,2), deși are suma 1 + (-2)+15 + 2 +12 + 6 = 34, nu este valid deoarece Greierașul are o grăunță când ajunge în celula (2,4), iar furnicuțele cer două.

Subiectul 4 – metoda Backtracking (3 p.)

a) Moș Crăciun are nevoie de m mașinuțe și apelează din nou la cei n spiriduși ai săi. El îl roagă pe fiecare spiriduș i $(1 \leq i \leq n)$ să-i spună care este numărul minim a_i și numărul maxim b_i de mașinuțe pe care ar vrea să le facă. Moș Crăciun ar vrea să îi pună pe spiriduși să facă cele m mașinuțe care-i trebuie pentru Crăciun, dar respectând opțiunile fiecărui spiriduș. Dacă vreți să primiți și voi una dintre mașinuțe, trebuie să-l ajutați pe Moș Crăciun scriind un program Python care să citească de la tastatură numerele m, n și opțiunile a_i și b_i ale fiecăruia dintre cei n spiriduși, după care să afișeze pe ecran toate modalitățile în care Moș Crăciun poate distribui producția de mașinuțe spiridușilor astfel încât spiridușul i să producă un număr de mașinuțe cuprins între a_i și b_i de mașinuțe sau mesajul "*Imposibil*" dacă nu există nicio modalitate de distribuție care să respecte cerințele precizate anterior. **(2.5 p.)**

Exemplu:
Pentru $m = 16, n = 3, a_1 = 1, b_1 = 6, a_2 = 0, b_2 = 7, a_3 = 4, b_3 = 8$ trebuie să fie afișate următoarele 20 de modalități de distribuție (nu neapărat în această ordine):

1 7 8
2 6 8
2 7 7
3 5 8
3 6 7
3 7 6
4 4 8
4 5 7
4 6 6
4 7 5
5 3 8
5 4 7
5 5 6
5 6 5
5 7 4
6 2 8
6 3 7
6 4 6
6 5 5
6 6 4

b) Modificați o singură instrucțiune din program astfel încât să fie afișate doar soluțiile în care spiridușul 1 și spiridușul 2 produc același număr de mașini. **(0.5 p.)**