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 **palindrom** care primeşte un număr variabil de cuvinte formate doar din litere mici ale alfabetului englez şi returnează informaţii despre cuvintele palindrom sub forma unui dicţionar de perechi **{cuvant palindrom: lista de litere}**. Cheia este cuvântul primit ca parametru dacă acesta este palindrom, iar lista de litere este formată din vocalele cuvântului dacă numărul vocalelor este mai mare decât numărul consoanelor, altfel lista va fi formată din consoanele cuvântului. Listele de litere vor fi sortate în ordine lexicografică. De exemplu, pentru apelul **palindrom ('asa', 'merem', 'palindrom')** funcţia va returna dicţionarul `{'asa': ['a'], 'merem': ['m', 'r']}` **(1.5 p.)**

b) Ştiind că matricea pătratică **m** este memorată sub forma unei liste de liste, înlocuiţi punctele de suspensie din instrucţiunea **numere = […]** cu o secvenţă de iniţializare (*list comprehension*) astfel încât, după executarea sa, lista să conţină pătratul elementelor aflate pe diagonala principală a matricei **m**. De exemplu, pentru matricea `m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]` trebuie ca lista **numere** să fie `[1, 25, 81]`. **(0.5 p.)**

c) Considerăm următoarea funcţie recursivă:

```
def f(lista):
    if len(lista) <= 2:
        return sum(lista)
    k = len(lista) // 3
    aux_1 = lista[:k]
    aux_2 = lista[k: 2*k]
    aux_3 = lista[2*k:]
    return f(aux_1) + f(aux_2) + f(aux_3)
```

Determinaţi complexitatea funcţiei apelată pentru o listă **L** formată din **n** numere întregi astfel: **f(L). (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: $O(n^2)$

Alice ar vrea să își schimbe parola la contul de email și are un șir de caractere preferat (format din caractere ASCII) de lungime *n* și un număr preferat *k*. Ea se gândește la următorul algoritm de construcție a unei parole din șirul ei de caractere preferat: șterge caractere din șir astfel încât șirul obținut după ștergere verifică următoarea proprietate - pentru orice două caractere aflate pe poziții consecutive în șir diferența dintre codurile lor ASCII (în modul) este mai mare sau egală cu *k*. Alice ar vrea ca parola să fie cel mai lung șir care se poate obține astfel din șirul ei preferat și numărul ei preferat *k* și vă roagă pe voi să scrieți un program Python care să citească șirul ei preferat și numărul *k* și să afișeze o parolă de lungime maximă care să verifice cerințele ei. În plus vă mai roagă îi spuneți și dacă soluția afișată este unică sau există mai multe astfel de parole de lungime maximă, afișând un un mesaj corespunzător: solutia optima este unica/ solutia optima nu este unica

Intrare de la tastatură	Ieșire pe ecran – nu este unică
`iepurasul_si_Alice_@.tara_minunilor` `15`	euas_s_Al@.tarau solutia optima nu este unica

Explicații*:* Codurile ASCII ale caracterelor din șir sunt
`105 101 112 117 114 97 115 117 108 95 115 105 95 65 108 105 99 101 95 64 46 116 97 114 97 95 109 105 110 117 110 105 108 111 114`
iar ale caracterelor din parolă sunt
`101 117 97 115 95 115 95 65 108 64 46 116 97 114 97 117`
Oricare două coduri consecutive din parolă diferă prin cel puțin 15 și nu există un alt șir de lungime mai mare cu această proprietate care se poate obține ștergând caractere din șirul inițial. Soluția optimă nu este unică, o altă soluție fiind de exemplu *euas_s_Al@.tar_u* (cu codurile `101 117 97 115 95 115 95 65 108 64 46 116 97 114 95 117`)

Subiectul 4 – metoda Backtracking (3 p.)

a) Petrişor ar vrea să îşi schimbe parolele şi are o mulţime de litere preferate **L** şi o mulţime **S** de simboluri (caractere care nu sunt litere) pe care ar vrea să le folosească în parole pentru a creşte siguranţa acestora . Pentru a îi fi mai uşor să le ţină evidenţa, el ar vrea să îşi construiască parole de aceeaşi lungime după un anumit tipar. Tiparul este un şir de caractere de lungime **n** format doar cu caracterele **'l'** şi **'s'** cu semnificaţia: dacă în tipar pe poziţia **i** este caracterul **'l'**, atunci în parolă pe poziţia **i** va fi o literă din mulţimea **L**, iar dacă în tipar pe poziţia **i** este caracterul **'s'**, atunci în parolă pe poziţia **i** va fi un simbol din mulţimea **S**. Mai mult, Petrişor şi-ar dori ca orice simbol din **S** şi orice literă din **L** să apară cel mult o dată în parolă. Scrieţi un program Python care să citească de la tastatură numărul **n**, tiparul **T** şi mulţimile **L** şi **S**, după afişează toate parolele care verifică cerinţele lui Petrişor sau mesajul "*Imposibil*" dacă nu există nicio parolă având proprietăţile cerute. **(2.5 p.)**

Exemplu: Pentru **n = 6**, tiparul **'lslsll'**, mulţimea **L** de litere **'a', 'b', 'c', 'D'** şi mulţimea **S** de simboluri **'@','.'** trebuie afişate următoarele 48 de parole (nu neapărat în această ordine):

a@b.cD	b@D.ca	c.D@ab
a@b.Dc	b.a@cD	c.D@ba
a@c.bD	b.a@Dc	D@a.bc
a@c.Db	b.c@aD	D@a.cb
a@D.bc	b.c@Da	D@b.ac
a@D.cb	b.D@ac	D@b.ca
a.b@cD	b.D@ca	D@c.ab
a.b@Dc	c@a.bD	D@c.ba
a.c@bD	c@a.Db	D.a@bc
a.c@Db	c@b.aD	D.a@cb
a.D@bc	c@b.Da	D.b@ac
a.D@cb	c@D.ab	D.b@ca
b@a.cD	c@D.ba	D.c@ab
b@a.Dc	c.a@bD	D.c@ba
b@c.aD	c.a@Db	
b@c.Da	c.b@aD	
b@D.ac	c.b@Da	

b) Modificaţi o singură instrucţiune din program astfel încât să fie afişate doar parolele care încep cu o vocală. Pentru exemplul anterior, aceste soluţii sunt cele scrise cu roşu. **(0.5 p.)**