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)

Bunicul vostru are N recipiente care conţin diverse cantităţi de apă la diferite temperaturi, respectiv recipientul i conține X_i litri de apă la temperatura Y_i. Bunicul vostru dorește să facă baie şi crede cu tărie faptul că apa în care se va spăla trebuie să aibă exact o temperatură T. De asemenea, bunicul consideră că este cu atât mai bine cu cât are mai multă apă la dispoziție. Din păcate, pe bunic nu îl ține spatele, așa că vă roagă pe voi să alegeţi inteligent ce cantitate de apă din fiecare recipient trebuie să turnaţi în cadă astfel încât bunicul vostru să se poată bucura de o baie plăcută! Temperatura apei nu este afectată de mediul exterior. La final, dacă aţi turnat $0 \leq C_i \leq X_i$ din recipiente, temperatura finala a apei din cadă va fi egală cu:

$$F = \frac{C_1 * Y_1 + C_2 * Y_2 + \cdots + C_N * Y_N}{C_1 + C_2 + \cdots + C_N}$$

Datele de intrare se vor citi de la tastatură, astfel:

- de pe prima linie două numere separate printr-un spațiu, respectiv numărul natural strict pozitiv N de recipiente şi numărul real T reprezentând temperatura dorită de bunicul vostru pentru apa din cadă;
- de pe următoarele N linii se vor citi câte două numere reale X_i şi Y_i cu semnificatia din enunţ (cantitatea de apă din recipientul i şi temperatura sa).

Programul Python trebuie să afişeze pe ecran cele N valori reale C_i, fiecare pe câte o linie, cu exact 3 zecimale. Se acceptă orice soluţie corectă.

Exemplu:

Date de intrare	Date de ieșire
3 40 50 20 72.5 50 38.72 90	50.000 72.500 5.500

Explicaţii: În cadă s-au turnat exact 50.000 + 72.5 + 5.5 = 128 litri, iar temperatura finală a apei este (50.000 * 20 + 72.5 * 50 + 5.5 * 90) / 128 = 5120 / 128 = 40 de grade. Nu există nicio altă variantă prin care să se toarne în cadă mai mult de 128 de litri şi apa să aibă exact temperatura de 40 de grade!

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) 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.)**