MODALITATEA DE DESFĂȘURARE A EXAMENULUI LA DISCIPLINA "PROGRAMAREA ALGORITMILOR" DIN SESIUNEA DE REEXAMINĂRI ȘI MĂRIRI DE NOTE 01 – 09 SEPTEMBRIE 2021

- **Examenul la disciplina "Programarea algoritmilor" se va desfășura în ziua de 02.09.2021, între orele 9^{00} și 14^{00}, astfel:**
  - 09^{00} – 09^{15}: efectuarea prezenței studenților la testul de laborator
  - 09^{15} – 10^{45}: desfășurarea testului de laborator
  - 10^{45} – 11^{00}: verificarea faptului că sursele trimise de către studenți au fost salvate pe platforma MS Teams
  - 11^{00} – 11^{30}: pauza
  - 11^{30}– 11^{45}: efectuarea prezenței studenților la examenul scris
  - 11^{45} – 13^{45}: desfășurarea examenului scris
  - 13^{45} – 14^{00}: verificarea faptului că fișierele trimise de către studenți au fost salvate pe platforma MS Teams

- Ambele probe se vor desfășura pe platforma MS Teams, iar pe tot parcursul desfășurării lor 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!

Precizări privind desfășurarea testului de laborator:

- Testul va conține 3 subiecte, iar un subiect poate să aibă mai multe cerințe.
- Rezolvarea unui subiect se va realiza într-un singur fișier sursă Python (.py), indiferent de numărul de cerințe, care va fi încărcat/atașat ca răspuns pentru subiectul respectiv.
- Numele fișierului sursă Python trebuie să respecte următorul șablon: *grupa_nume_prenume_subiect.py*. 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.py.
- La începutul fiecărui fișier sursă Python se vor scrie, sub forma unor comentarii, următoarele informații: numele și prenumele studentului, grupa sa și enunțul subiectului rezolvat în fișierul sursă respectiv (copiat direct din MS Teams). Dacă un student nu reușește să rezolve deloc un anumit subiect, totuși va trebui să încarce/atașeze un fișier sursă Python cu informațiile menționate anterior!

- Toate rezolvările (fișierele sursă Python) trimise de către studenți vor fi verificate din punct de vedere al similarității folosind un software specializat, iar eventualele fraude vor fi sancționate conform Regulamentului de etică și profesionalism al FMI (http://old.fmi.unibuc.ro/ro/pdf/2015/consiliu/Regulament_etica_FMI.pdf).

Precizări privind desfășurarea examenului scris:

- 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.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.pdf*.

Subiectul 1 – limbajul Python – 3 p.

a) Scrieţi o funcţie *note* care primeşte un număr variabil de liste cu elemente numere naturale, fiecare reprezentând lista notelor elevilor dintr-o clasă la un test şi returnează un dicţionar de perechi de forma {*numar_promovati : lista_de_liste*}, unde cheia *numar_promovati* reprezintă un număr, iar *lista_de_liste* este o listă nevidă care are ca elemente toate listele primite ca parametru care au numărul de elemente mai mari sau egale ca 5 egal cu *numar_promovati*.

De exemplu, pentru apelul *note*([5, 4, 2, 7, 10], [6, 7, 8, 10, 3], [10, 7, 4, 10, 9], [5, 6, 8, 4, 1], [5, 5, 6, 10, 7, 9]) funcţia trebuie să furnizeze dicţionarul {3: [[5, 4, 2, 7, 10], [5, 6, 8, 4, 1]], 4: [[6, 7, 8, 10, 3], [10, 7, 4, 10, 9]], 6: [[5, 5, 6, 10, 7, 9]]}, deoarece listele [5, 4, 2, 7, 10] şi [5, 6, 8, 4, 1]] au 3 elemente mai mari sau egale cu 5, listele [[6, 7, 8, 10, 3], [10, 7, 4, 10, 9] au 4 astfel de elemente, iar lista [5, 5, 6, 10, 7, 9] are 6 elemente mai mari sau egale cu 5. **(1.5 p.)**

b) Ştiind că *lista_cuvinte* este o listă cu elementele cuvinte (de tip str) şi *lista_sufixe* este o listă de cuvinte de lungime 2, înlocuiţi punctele de suspensie din instrucţiunea *lista_rez* = [...] cu o secvenţă de iniţializare (*list comprehension*) astfel încât, după executarea sa, lista *lista_rez* să conţină cuvintele din *lista_cuvinte* care au sufixul format cu ultimele două litere în lista *lista_sufixe* De exemplu, dacă *lista_cuvinte* = ["acasa", "masa", "este", "scaun", "perete", "dulap"] şi *lista_sufixe* = ["sa", "te"] atunci *lista_rez* va avea ca elemente cuvintele 'aca**sa**', 'ma**sa**', 'es**te**', 'pere**te**'. **(0.5 p.)**

c) Considerăm următoarea funcţie recursivă (se presupune importat modulul random):

```
def f(lista, p, u):
    if u - p <= 2:
        return sum(lista[p:u + 1])
    k = (u - p + 1) // 3
    x = random.randint(0, 3)
    if x == 1:
        rez =  f(lista, p, p + k)
    elif x == 2:
        rez =  f(lista, p + k + 1, p + 2 * k - 1)
    else:
        rez =  f(lista, p + 2 * k, u)
    for i in range(p,u+1):
        rez += lista[i]
    return rez
```

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: $\mathcal{O}(n \log_2 n)$

La ora de sport, profesorul vrea să execute exerciții de gimnastică cu grupe de câte 2 elevi, dar pentru a putea realiza acest lucru trebuie ca valoarea absolută a diferenței dintre înălțimile celor 2 elevi dintr-o grupă să fie strict mai mică decât un număr natural *h*. Scrieți un program Python care citește de la tastatură numele și înălțimile a *n* elevi și afișează pe ecran, în forma indicată în exemplu, numărul maxim de grupe formate din câte 2 elevi care se pot forma respectând condiția indicată anterior, precum și numele elevilor din grupele respective. Evident, un elev poate să facă parte din cel mult o grupă! Înălțimile tuturor elevilor și diferența *h* sunt exprimate în centimetri. Nu contează ordinea în care se vor afișa grupele de elevi și nici ordinea numelor elevilor dintr-o grupă. Dacă nu se poate forma nicio grupă de 2 elevi cu proprietatea cerută se va afișa un mesaj corespunzător.

Exemplu:

Date de intrare	Date de ieșire
n = 8 h = 10 Numele și înălțimile elevilor: Popescu Ion 172 Mihai Ana 162 Popescu Dana 190 Ionescu Ion 181 Georgescu Ioana 170 Dumitrescu George 188 Constantinescu Radu 165 Georgescu Anca 210	3 Popescu Ion, Georgescu Ioana Mihai Ana, Constantinescu Radu Ionescu Ion, Dumitrescu George

Explicații: Se pot forma maxim 3 grupe de câte 2 elevi cu proprietatea că valoarea absolută a diferenței dintre înălțimile lor este strict mai mică decât 10 centimetri. Soluția nu este unică, o altă soluție corectă obținându-se, de exemplu, înlocuind grupa *Ionescu Ion, Dumitrescu George* cu grupa *Ionescu Ion, Popescu Dana*.

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

După o nouă plimbare lungă prin parc cu stăpâna sa, cățelușa Laika se află din nou în fața unei mari provocări: trebuie iar să urce cele *n* trepte până la ușa apartamentului în care locuiește. De data aceasta ea mai are însă energie și poate să sară de pe o treaptă i direct pe una dintre treptele *i*+1, *i*+2, ..., *i*+*k*, unde *k* este un număr natural nenul dat. Pentru că treptele au fost spălate și Laika vine după o joacă prin noroi, pentru fiecare treaptă *i* (unde $i \in \{1,...,n\}$) pe care calcă Laika stăpâna trebuie să plătească o taxă t_i (număr natural nenul). Ajutați-o pe Laika, scriind un program Python care să afișeze o modalitate de a urca treptele începând de la baza scării (considerată treapta 0, pentru care taxa este t_0 = 0) la treapta *n* pentru care suma totală pe care trebuie să o plătească stăpâna să fie minimă. Se dau *n*, *k* și taxele pentru cele *n* trepte t_1, t_2, ..., t_n.

Intrare de la tastatură	Ieșire pe ecran
`7 2` `1 2 5 1 1 6 4`	`taxa totala 8 pentru traseul cu scările` `0 2 4 5 7`

Explicații: Avem *n*=7, deci scara are 7 trepte numerotate 1, 2, ..., 7 (pe lângă baza scării care se consideră treapta 0 și pentru care nu se plătește taxă) și *k*=2, deci Laika poate sări de pe o treaptă *i* pe treapta *i*+1 sau pe treapta *i*+2. Traseul optim este format cu scările 0, 2, 4, 5, 7 (traseul trebuie să înceapă de la baza scării, adică treapta 0 și să se termine cu scara 7), taxa care trebuie plătită fiind $t_2 + t_4 + t_5 + t_7 = 2 + 1 + 1 + 4 = 8$.

Subiectul 4 – metoda Backtracking (3 p.)

a) Un număr natural se numește *i,j-mărginit* ($0 \leq i \leq j \leq 9$) dacă orice cifră a sa este cuprinsă între *i* și *j*. De exemplu, numărul 27338 este 2,8-*mărginit*, iar numărul 4122042 este 0,4-*mărginit*. Scrieți un program Python care să citească de la tastatură numerele naturale *i* și *j*, după care afișează toate numerele naturale *i,j-mărginite* formate din cifre distincte sau un mesaj corespunzător dacă nu există niciun astfel de număr. **(2.5 p.)**

Exemplu:
Pentru $i = 2$ și $j = 5$ trebuie afișate următoarele 64 de numere (nu neapărat în această ordine):

2	3	4	5
23	32	42	52
234	324	423	523
2345	3245	4235	5234
235	325	425	524
2354	3254	4253	5243
24	34	43	53
243	342	432	532
2435	3425	4325	5324
245	345	435	534
2453	3452	4352	5342
25	35	45	54
253	352	452	542
2534	3524	4523	5423
254	354	453	543
2543	3542	4532	5432

b) Precizați cum ar trebui modificată o singură instrucțiune din program astfel încât să fie afișate doar numerele *i,j-mărginite* care au prima cifră egală cu *j*. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. **(0.5 p.)**