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) Înlocuiţi punctele de suspensie din instrucţiunea **numere = […]** cu o secvenţă de iniţializare (*list comprehension*) astfel încât, după executarea sa, lista să conţină numerele naturale formate din exact două cifre care nu sunt pătrate perfecte şi nici divizibile cu 7. **(0.5 p.)**

c) Considerăm următoarea funcţie recursivă:

```
def f(lista):
    if len(lista) <= 2:
        return min(lista)
    k = len(lista) // 2
    aux_1 = lista[:k]
    aux_2 = lista[k+1:]
    return min(f(aux_1), f(aux_2), lista[k])
```

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

O balanță veche s-a defectat și acum se echilibrează nu doar pentru două obiecte având aceeași greutate, ci pentru orice două obiecte cu proprietatea că modulul diferenței dintre greutățile lor este mai mic sau egal decât un număr real *g*. Scrieți un program Python care citește de la tastatură un număr natural n, un număr real g și greutățile a *n* obiecte, după care afișează pe ecran numărul maxim de perechi de obiecte care echilibrează balanța defectă, precum și perechile respective, știind că orice obiect poate să facă parte din cel mult o pereche. Fiecare pereche afișată trebuie să fie de forma *x* + *y*, unde *x* și *y* sunt numerele de ordine ale celor două obiecte din pereche (obiectele sunt numerotate începând de la 1). Greutățile tuturor obiectelor și diferența *g* sunt exprimate prin numere reale strict pozitive, reprezentând grame. Nu contează ordinea în care se vor afișa perechile de obiecte pe ecran și nici ordinea numerelor de ordine ale obiectelor dintr-o pereche.

Exemplu:

Date de intrare	Date de ieșire
10 8.5 21.25 12 6.05 20.7 23.8 22 33.25 21 48.15 62.20	3 3 + 2 10 + 4 1 + 8

Explicații: Avem n = 10 și g = 8.5. Se pot forma maxim 3 perechi de obiecte care pot echilibra balanța defectă. Soluția nu este unică, o altă soluție corectă obținându-se, de exemplu, înlocuind perechea 1 + 6 cu perechea 1 + 5.

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) Un număr natural se numeşte *echilibrat* dacă valoarea absolută a diferenţei dintre oricare două cifre aflate pe poziţii consecutive şi, respectiv, între prima şi ultima sa cifră este 0 sau 1. De exemplu, numerele 5456 şi 1232 sunt echilibrate, iar numerele 5443, 5446 şi 1233 nu sunt echilibrate. Scrieţi un program Python care citeşte de la tastatură un număr natural m ($2 \leq m \leq 20$) şi afişează toate numerele naturale *echilibrate* formate din exact m cifre, precum şi numărul acestora. **(2.5 p.)**

Exemplu:
Pentru $m = 4$ trebuie afişate următoarele numere (nu neapărat în această ordine):

1000	3232	5456	7767
1001	3233	5544	7776
1010	3234	5545	7777
1011	3322	5554	7778
1012	3323	5555	7787
1100	3332	5556	7788
1101	3333	5565	7876
1110	3334	5566	7877
1111	3343	5654	7878
1112	3344	5655	7887
1121	3432	5656	7888
1122	3433	5665	7898
1210	3434	5666	8767
1211	3443	5676	8777
1212	3444	6545	8778
1221	3454	6555	8787
1222	4323	6556	8788
1232	4333	6565	8789
2101	4334	6566	8877
2111	4343	6567	8878
2112	4344	6655	8887
2121	4345	6656	8888
2122	4433	6665	8889
2123	4434	6666	8898
2211	4443	6667	8899
2212	4444	6676	8987
2221	4445	6677	8988
2222	4454	6765	8989
2223	4455	6766	8998
2232	4543	6767	8999
2233	4544	6776	9878
2321	4545	6777	9888
2322	4554	6787	9889
2323	4555	7656	9898
2332	4565	7666	9899
2333	5434	7667	9988
2343	5444	7676	9989
3212	5445	7677	9998
3222	5454	7678	9999
3223	5455	7766	Nr. solutii:159

b) Precizaţi cum ar trebui modificată o singură instrucţiune din program astfel încât să fie afişate doar numerele *echilibrate* formate din m cifre şi care sunt numere pare. Pentru exemplul anterior, aceste soluţii sunt cele scrise cu roşu. **(0.5 p.)**