# MODALITATEA DE DESFĂȘURARE A TESTULUI DE LABORATOR LA DISCIPLINA "PROGRAMAREA ALGORITMILOR"

- Testul de laborator la disciplina "Programarea algoritmilor" se va desfășura în ziua de **08.01.2022**, între orele $9^{30}$ și $12^{00}$, astfel:
  - **$09^{30}$ – $10^{00}$**: efectuarea prezenței studenților
  - **$10^{00}$ – $11^{30}$**: desfășurarea testului
  - **$11^{30}$ – $12^{00}$**: verificarea faptului că sursele trimise de către studenți au fost salvate pe platformă

- Testul se va desfășura pe platforma MS Teams, iar pe tot parcursul desfășurării lui 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!

- 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. 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).

**Subiect 1**

**[4 p.]** Fișierul text *numere.in* conține, pe mai multe linii, numere naturale despărțite prin spații. Definim *indicele binar* al unui număr natural *n* ca fiind perechea (*x*, *y*), unde *x* reprezintă numărul biților nenuli din reprezentarea binară a lui *n*, iar *y* reprezintă numărul biților nuli din reprezentarea sa binară. De exemplu, numărul 133 are reprezentarea binară 10000101, deci indicele său binar este (3, 5). Să se scrie în fișierul text *numere.out* numerele din fișierul *numere.in* grupate în funcție de indicii lor binari, conform modelului din exemplul de mai jos. Grupele de numere vor fi scrise în ordinea descrescătoare a numărului de numere pe care le conțin, iar în caz de egalitate se vor scrie în ordinea lexicografică a indicilor lor binari. În cadrul fiecărei grupe numerele vor fi ordonate crescător. Fiecare număr va fi scris o singură dată în fișierul de ieșire.

**Exemplu:**

| **numere.in** | **numere.out** |
|---|---|
| 101 133 102<br>224 152 10<br>133 133 101 888<br>7 12 543 120 137<br>10 108 108 102 | (3, 5): 133,137,152,224<br>(4, 3): 101,102,108,120<br>(2, 2): 10,12<br>(6, 4): 543,888<br>(3, 0): 7 |

**Subiectul 2**

Se dă fișierul **"matrice.in"** cu următoarea structură: pe linia $i$ se află separate prin câte un spațiu $n$ numere naturale reprezentând elementele de pe linia $i$ a unei matrice, ca în exemplul de mai jos. Este cunoscut faptul că în fișier se află $n * n$ elemente numere naturale, unde $n$ este un număr natural impar $> 2$.

**a) [0,25p]** Să se scrie o funcție **citire_matrice** care citește datele din fișierul **"matrice.in"** și returnează o matrice de dimensiune $n \times n$ formată din aceste numere.

**b) [1,5p]** Să se scrie o funcție care primește ca parametru matricea și returnează matricea bordată după următoarele reguli:

- se va adăuga o coloană nouă la final (pe poziția $n$) care va avea pe poziția $k$ **suma** valorilor de pe **linia** $k$
- se va adăuga o linie nouă la final (poziția $n$) care va avea pe poziția $k$ **suma** valorilor de pe **coloana** $k$

**c) [1,25p]** Se consideră matricea citită la punctul **a)**, peste care se aplică funcția de la punctul **b)**. Să se parcurgă matricea mai întâi pe diagonala principală, apoi pe diagonala secundară și, în final, restul elementelor care **nu** aparțin diagonalelor (parcurgerea se face pe linii de sus în jos și de la stânga la dreapta) și se afișează elementele în fișierul **"matrice.out"**.

| **matrice.in** | **după bordare** | **matrice.out** |
|---|---|---|
| 7 8 9<br>6 1 2<br>5 4 3 | 7 8 9 24<br>6 1 2 9<br>5 4 3 12<br>18 13 14 45 | 7 1 3 45 24 2 4 18 8 9 6 9 5 12 13 14 |

**Subiect 3**

Se consideră o rețea în plan formată din puncte unite prin linii numite legături (muchii). Fiecare punct are coordonatele întregi, iar o legătură are asociată o culoare (un șir de caractere fără spații reprezentând numele culorii, de exemplu: roșu, verde, albastru) și o etichetă (un șir care poate conține spații). Un punct cu coordonatele x și y va fi notat (x,y). Se consideră fișierul text *legaturi.in* care conține informații despre o astfel de rețea, fiecare linie conținând informații despre o legătură sub forma:

***[x1,y1] [x2,y2] culoare eticheta***

unde [x1,y1] și [x2,y2] sunt punctele între care există legătură, iar ***culoare*** și ***etichetă*** reprezintă culoarea și eticheta asociate legăturii. Vom spune că legătura este între punctele [x1,y1] și [x2,y2], vom numi punctele [x1,y1] și [x2,y2] capetele legăturii (legătura este neorientată și bidirecțională) și cele două puncte sunt vecine. Un exemplu de fișier de acest tip este următorul:

**legaturi.in**

```
[1,2] [1,3] rosu legatura 1
[1,2] [1,4] albastru legatura 2
[1,3] [2,6] rosu legatura 3
[2,6] [2,7] albastru legatura 4
[2,7] [3,8] rosu legatura 5
```

**a) [2 p.]** Să se memoreze datele din fișier într-o singură structură astfel încât să se **răspundă eficient** la cerințele de la punctele următoare (interogarea și modificarea informațiilor despre o legătură dintre două puncte date, determinarea vecinilor unui punct).

**b) [1 p.]** Scrieți o funcție **insereaza_legatura** care primește 5 parametri:

- în primul parametru se transmite structura în care s-au memorat datele la cerința a)
- următorii 2 parametri sunt tupluri cu 2 elemente reprezentând capetele unei legături
- ultimii 2 parametri sunt șiruri de caractere reprezentând culoarea și eticheta unei legături.

Dacă există deja o legătură în rețea între punctele primite ca parametru funcția va returna **False**, altfel funcția va adăuga această legătură în rețea (modificând structura trimisă ca parametru) și va returna **True**. Să se apeleze funcția pentru punctele (1,3) și (2,7), culoarea negru și eticheta "legatura noua" și să se afișeze legăturile memorate în structura obținută în același format în care s-au dat și datele în fișier (nu contează ordinea în care se vor afișa legăturile).

**ieșire**

```
[1,2] [1,3] rosu legatura 1
[1,2] [1,4] albastru legatura 2
[1,3] [2,6] rosu legatura 3
[2,6] [2,7] albastru legatura 4
[2,7] [3,8] rosu legatura 5
[1,3] [2,7] negru legatura noua
```

**c) [1 p.]** Scrieți o funcție **vecini** care primește ca parametri (în această ordine): structura în care s-au memorat datele la cerința a) și un număr variabil de tupluri cu 2 elemente reprezentând puncte din rețea. Funcția va returna o listă cu acele puncte *p* din rețea cu proprietatea că există legătură de la *p* la orice punct primit ca parametru (vecinii comuni pentru punctele primite ca parametru). Punctele din lista returnată vor fi ordonate descrescător după a doua coordonată. Apelați funcția pentru punctele (2,7) și (1,2) și afișați rezultatul obținut.

**ieșire**

```
[(3, 8), (1, 3)]
```