Exemplu subiect test.

În fișierul text *matricerara.csv* este salvată o matrice rară. Pe fiecare linie sunt informații despre un element, astfel: index linie (*int*), index coloană (*int*) și valoare element (*double*). Valorile sunt despărțite prin virgulă. Cei doi indecși arată poziția elementului în matrice.
Să se scrie o aplicație care să îndeplinească următoarele cerințe:
1) Să se construiască o clasă *Element* care să permită stocarea informațiilor despre un element conform structurii de mai sus. Clasa va avea implementate:
- constructori (de inițializare și *default*), metode de acces, *toString*();
- testare egalitate după index linie, index coloană și valoare (două elemente sun egale dacă au aceeași poziție și valori egale);
- comparabilitate între elemente după valoare.
Punctaj: 2 puncte
2) Să se citească matricea rară într-o listă (*List<Element>*) și să se afișeze la consolă numărul de elemente negative
Punctaj: 2 puncte
3) Să se afișeze mediile pe coloane ale matricei, astfel:
index_coloana:valoare
...
index_coloana:valoare
Punctaj: 3 puncte (prin utilizare colectori), 1 punct (fără colectori)
4) Să se salveze în fișierul binar *diagonal.dat* elementele de pe diagonala principală
Punctaj: 1 punct
5) Să se citească elementele de pe diagonala principală din fișierul *diagonal.dat* și să se afișeze la consolă.
Punctaj: 1 punct

Observații
Nu se acordă punctaj pentru programele cu erori de sintaxă sau erori în execuție.
Punctajul se acordă dacă rezultatele sunt corecte și furnizate conform cerințelor.

2.

În fișierul *puncte.csv* sunt salvate punctele unor figuri geometrice. Pe fiecare linie sunt informații despre un punct, astfel: eticheta figurii din care face parte punctul (*String*), eticheta punctului (String), coordonată pe axa *Ox* (*double*) și coordonată pe axa *Oy* (*double*). Valorile sunt despărțite prin virgulă.
Să se scrie o aplicație care să îndeplinească următoarele cerințe:
1) Să se construiască o clasă *Punct* care să permită stocarea informațiilor despre un punct conform structurii de mai sus. Clasa va avea implementate:
- constructori (de inițializare și *default*), metode de acces, *toString*();
- implementarea unei metode *distanta*() care să calculeze și să returneze distanța față de origine ($\sqrt{x^2 - y^2}$, unde *x* și *y* sunt coordonatele punctului);
- comparabilitate între elemente după distanța față de origine.
Punctaj: 2 puncte
2) Să se citească punctele figurilor într-o listă (*List<Punct>*) și să se afișeze la consolă numărul de puncte
Punctaj: 2 puncte
3) Să se afișeze numărul de puncte pentru fiecare figură, astfel:
eticheta_figura:numar_puncte
...
eticheta_figura:numar_puncte
Punctaj: 3 puncte (prin utilizare colectori), 1 punct (fără colectori)
4) Să se salveze în fișierul text *distante.csv* distanțele punctelor față de origine, calculate prin metoda cerută la punctul 1, sortate descrescător, astfel:
eticheta_figura,eticheta_punct,distanta
...
eticheta_figura,eticheta_punct,distanta
Punctaj: 2 puncte

Observații
Nu se acordă punctaj pentru programele cu erori de sintaxă sau erori în execuție.
Punctajul se acordă dacă rezultatele sunt corecte și furnizate conform cerințelor.

3.
În fișierul *jurnal.csv* sunt salvate notele contabile pentru exercițiul curent (luna curentă). Pe fiecare linie sunt informații despre o operațiune contabilă, astfel: data operațiunii (*Date* - în format dd.MM.yyyy), cont debitor (*int* - simbolul contului), cont creditor (*int*), suma (*double*). Valorile sunt despărțite prin virgulă.
Exemplu
01.04.2022,5311,5121,10000
03.04.2022,401,5121,25000
02.04.2022,300,401,25000
04.04.2022,5121,411,70000
04.04.2022,411,701,70000
Să se scrie o aplicație care să îndeplinească următoarele cerințe:
1) Să se construiască o clasă *NotaContabila* care să permită stocarea informațiilor despre o operațiune contabilă conform structurii de mai sus. Clasa va avea:
- constructori (de inițializare și *default*), metode de acces, *toString*();
- testare egalitate după cont debitor, cont creditor și suma;
- comparabilitate între operațiuni după dată.
Punctaj: 2 puncte
2) Să se citească operațiunile contabile într-o listă (*List<NotaContabila>*) și să se afișeze la consolă rulajul total (se obține prin totalizarea sumelor tuturor operațiunilor)
Punctaj: 2 puncte
3) Să se afișeze rulajul total pentru fiecare cont, astfel:
simbol_cont:rulaj
...
simbol_cont:rulaj
Punctaj: 3 puncte (prin utilizare colectori), 1 punct (fără colectori)
4) Să se salveze în fișierul text *fisa.csv* fișa unui cont specificat, astfel:
data,tip_operațiune,cont_corespondent,suma
...
data,tip_operațiune,cont_corespondent,suma
unde *tip_operatiune* poate fi "debitare" sau "creditare" după cum contul specificat este cont debitor sau creditor. Contul pentru care se listează fișa este citit de la tastatură sau se inițializează explicit. Fișa este generată în ordine cronologică după dată.
Exemplu pentru 5121:
01.04.2022,Creditare,5311,10000
03.04.2022,Creditare,401,25000
04.04.2022,Debitare,411,70000

Punctaj: 2 puncte

Observații
Nu se acordă punctaj pentru programele cu erori de sintaxă sau erori în execuție.
Punctajul se acordă dacă rezultatele sunt corecte și furnizate conform cerințelor.

4.
În fișierul *examene.csv* sunt salvate examenele susținute la nivelul unei facultăți în ultima sesiune. Pe fiecare linie sunt informații despre un examen, astfel: data examenului (*Date* - în format dd.MM.yyyy), profesor titular (*String*), nume disciplină (*String*), număr studenți înscriși în catalog (*int*), număr studenți examinați (*int*) . Valorile sunt despărțite prin virgulă.
Să se scrie o aplicație care să îndeplinească următoarele cerințe:
1) Să se construiască o clasă *Examen* care să permită stocarea informațiilor despre un examen conform structurii de mai sus. Clasa va avea:
- constructori (de inițializare și *default*), metode de acces, *toString*();
- implementarea unei metode *absenteism*() care să calculeze și să returneze rata absenteismului (procentul numărului de studenți înscriși în catalog dar neexaminați raportat la numărul de studenți înscriși în catalog);
- comparabilitate între examene după număr de studenți examinați.
Punctaj: 2 puncte
2) Să se citească examenele într-o listă (*List<Examen>*) și să se afișeze la consolă numărul total de examene.
Punctaj: 2 puncte

3) Să se afișeze absenteismul mediu pe fiecare disciplină, astfel:
nume_disciplina:absenteism_mediu
...
nume_disciplina:absenteism_mediu
Punctaj: 3 puncte (prin utilizare colectori), 1 punct (fără colectori)
4) Să se salveze în fișierul text *discipline.csv* examenele susținute la fiecare disciplină, astfel:
nume_disciplina_1
data_examen,nume_titlular,numar_examinati
...
data_examen,nume_titlular,numar_examinati
nume_disciplina_2
data_examen,nume_titlular,numar_examinati
...
data_examen,nume_titlular,numar_examinati
...
Punctaj: 2 puncte

Observații
Nu se acordă punctaj pentru programele cu erori de sintaxă sau erori în execuție.
Punctajul se acordă dacă rezultatele sunt corecte și furnizate conform cerințelor.