Operatii extra laborator SDA 28.05.2025

TAD Matrice
// creaza un iterator bi-direcțional asupra tuturor elementelor matricei pe o anumită coloană (indiferent dacă acestea sunt sau nu NULL_TELEM).
// aruncă excepție în cazul în care coloana nu este o coloană validă
IteratorMatrice iterator(coloana int) const;
Obs: Va trebui implementată clasa IteratorMatrice (având funcționalitatea descrisă anterior), de asemenea.

TAD Colecție
// păstrează în colecția curentă numai elementele care apar în colecția *b* astfel:
// daca un element apare de mai multe ori în ambele colecții, acesta va fi păstrat de un număr de ori egal cu minimul dintre numărul de aprații în colecția curentă și numărul de aparații în colecția *b*, transmisă ca argument
void intersecție (const Colecție & b);

TAD Mulțime
// returnează diferența dintre valoarea maximă și cea minimă (presupunem valori întregi)
// dacă mulțimea este vidă, se returnează -1
int diferențaMaxMin() const;

TAD Listă
// păstrează în listă numai elementele care respectă condiția dată
void filtreaza(Condiția cond);
Condiția este o funcție care primește ca parametru un TElem și returnează adevărat sau fals (adevărat dacă elementul respectă condiția și fals, altfel)
Obs: adăugați următorul typedef la începutul fișierului Listă.h
typedef bool (*__Condiție__)(TElem);

TAD ListăOrdonată
// păstrează în listă numai elementele care respectă condiția dată
void filtreaza(Condiția cond);
Condiția este o funcție care primește ca parametru un TElem și returnează adevărat sau fals (adevărat dacă elementul respectă condiția și fals, altfel)
Obs: adăugați următorul typedef la începutul fișierului Listă.h
typedef bool (*__Condiție__)(TElem);

TAD Dicționar
// găsește și returnează cheia minimă a dicționarului
// Dacă dicționarul este vid, se returnează NULL_TCHEIE
TCheie cheieMinima() const;
Obs: Pentru a avea NULL_TCHEIE, se adaugă la începutul Dictionar.h următoarea definiție:
#define NULL_TCHEIE 0

TAD DicționarOrdonat
// găsește și returnează cheia minimă a dicționarului
// Dacă dicționarul este vid, se returnează NULL_TCHEIE
TCheie cheieMinima() const;
Obs: Pentru a avea NULL_TCHEIE, se adaugă la începutul Dictionar.h următoarea definiție:
#define NULL_TCHEIE 0

TAD Multidicționar
// găsește și returnează cheia minimă a multidicționarului
// Dacă multidicționarul este vid, se returnează NULL_TCHEIE
TCheie cheieMinima() const;
Obs: Pentru a avea NULL_TCHEIE, se adaugă la începutul MD.h următoarea definiție:
#define NULL_TCHEIE 0

TAD MultidicționarOrdonat
// găsește și returnează cheia minimă a multidicționarului
// Dacă multidicționarul este vid, se returnează NULL_TCHEIE
TCheie cheieMinima() const;
Obs: Pentru a avea NULL_TCHEIE, se adaugă la începutul MD.h următoarea definiție:
#define NULL_TCHEIE 0

TAD CoadăCuPriorități
Transformarea într-o CoadăCuPriorități cu capacitate fixă.
Efectuați următoarele modificări în proiect:

- Constructorul clasei CoadăCuPriorități primește ca parametru un număr întreg reprezentând capacitatea. În cazul în care acest număr este 0 sau negativ, se aruncă excepție.
- Operația de adăugare aruncă excepție în cazul în care coada este plină.
- Adăugați o operație estePlină (similar cu esteGoală/vidă) care returnează true dacă coada este plină și false în caz contrar.

Obs: Aceste modificări se vor efectua pe o copie a proiectului, pentru a vă asigura că testele vor rula în continuare pe proiectul inițial.