Proiectul I

Cerințe globale

Următoarele cerințe sunt valabile pentru toate proiectele:

- Comentarii în cod care să ofere detalii despre modul de implementare.
- Indentarea și spațierea adecvată a codului.
- Utilizarea unei convenții de denumire a variabilelor și a metodelor.
- Separarea declarației de implementare a clasei/claselor. Declararea se va face în .h (header) și implementare în .cpp.
- Supraîncărcarea operatorilor de citire și afișare (<< și >>).
- Implementatea constructorilor cu parametri, fără parametri, de copiere și supraîncărcarea operatorului =.
- Implementarea destructorului.
- Alocarea dinamică a memoriei.

Alte observații

- Funcționalitățile implementate vor fi testate folosing assert.
- Fiecare clasă va suporta număr arbitrar de intrări (pentru cele în care se pot adauga elemente).
- Pentru punctajul maxim implementările trebuie să fie cât mai eficiente din punct de vedere timp și spațiu.
- Folosirea operatorilor << și >> în cod, cu excepția supraîncărcarii operatorilor de citire și afișare, nu este permisă.
- Folisirea utilităților oferite de STL nu este permisă.
- În cazul parcurgerilor rezultatul întors va fi un arrau care conține elementele în ordinea de parcurgere (e.g. primul element parcurs pe prima poziție în vector, al doilea element pe a doua poziție în vector ș.a.m.d).
- Fiecare clasă trebuie sa includă doar funcționalitățile enumerate în cerințe. Dacă se consideră necesare alte metode acestea trebuie sa fie declarate ca private.

- Întârzierea cu o zi scade cu 20p nota maximă obținută. Dacă se trimite proiectul între orele 00:00 - 23:59 ale primei zi după deadline se vor scădea 20p. Dacă se trimite proiectul între orele 00:00 - 23:59 ale zilei 2 după deadline se vor scădea 40p. Orice trimitere după mai mult de două zile va face ca proiectul să nu mai fie luat în considerare.
- Proiectul trebuie urcat pe github. Dacă proiectul este într-un repositor privat, adăugățimă la colaboratori. Dacă proiectul este într-un repositor public, trimiteți-mi un email cu link către proiect. Link-ul de github se poate trimite și după deadline.
- DEADLINE: 22 martie 2020, ora 12:00

1 Set (mulțime)

Implementați clasa **Set** (mulțime) care să ofere următoarele funcționalități:

- Fiecare element sa apară o singură dată (operația care va realiza această funcționalitate va fi implementată cât mai eficient).
- Inserarea, ștergerea și căutarea unui element.
- Supraîncărcarea operatorului [ ] pentru accesarea elementului de pe poziția i.
- Supraîncărcarea operatorului + care să efectueze reuniunea a două mulțimi (fără duplicate).
- Supraîncărcarea operatorilor $<$ și $>$ care să compare cardinalul a două mulțimi.
- Supraîncărcarea operatorului * pentru înmulțirea cu un scalar.
- Obținerea sumei elementelor din mulțime.
- Obținerea elementelor pare, respectiv impare din mulțime (sub formă de Set).
- Obținerea numărul de elemente din mulțime.

2 Graf

Implementați clasa **Graf** care să ofere următoarele funcționalități:

- Parcurgerea în lățime și lungime pornind dintr-un nod dat.
- Distanța dintre două noduri date.
- Supraîncărcarea operatorului [ ] care să întoarcă lista de adiacentă a unui nod sub formă de array.
- Supraîncărcarea operatorilor $<$ și $>$ care să compare numărul de noduri și numărul de muchii a două grafuri (Fie $G_1 = (n_1, m_1)$ și $G_2 = (n_2, m_2)$, unde n_i reprezintă numărul de noduri din graful i, iar m_i reprezintă numărul de muchii din graful i. $G_1 > G_2$ dacă $n_1 > n_2$ sau $n_1 == n_2$ && $m_1 > m_2$).
- Stabilirea dacă graful este arbore.
- Determinarea numărului de componente conexe din graf.
- Adăugarea unei muchii la graf.
- Obținerea numărului de noduri, respectiv muchii, ale grafului.

3 Matrice

Implementați clasa **Matrice** care să ofere următoarele funcționalități:

- Supraîncărcarea operatorului + pentru adunarea a doua matrici.
- Supraîncărcarea operatorului [ ] pentru a obținerea liniei i.
- Supraîncărcarea operatorului - pentru scăderea a două matrici.
- Supraîncărcarea operatorului * pentru înmulțirea a doua matrici și înmulțirea cu un scalar.
- Calcularea determinantului matricei.
- Determinarea inversabilității matricei.
- Obținerea unei matrici de ordin inferior prin eliminarea unei coloane, unei linii sau unei linii și unei coloane.
- Obținerea numărului de linii, respectiv coloane.

4 Polinoame

Implementați clasa **Polinom** care să ofere următoarele funcționalități:

- Calcularea valorii polinomului într-un anumit punct.
- Supraîncărcarea operatorului + pentru adunarea a două polinoame.
- Supraîncărcarea operatorului [ ] pentru obținerea termenului cu gradul i.
- Supraîncărcarea operatorului * pentru înmulțirea a două polinoame și înmulțirea cu scalar.
- Supraîncărcarea operatorului / pentru împărțirea a două polinoame.
- Adăugarea și eliminarea unui termen de grad i.
- Obținerea gradului polinomului.

5 Coadă cu Priorități

Implementați clasa **PQueue** (coadă de numere întregi cu priorități) care va avea următoarele funcționalități:

- Adăugarea și eliminarea elementelor în/din coadă.
- Obținerea numărului de elemente din coadă.
- Supraîncărcarea operatorului + pentru fuziunea a două cozi.
- Supraîncărcarea operatorilui ++ care va crește cu 1 prioritățile elementelor din coadă.

- Supraîncărcarea operatorilui care va scădea cu 1 prioritățile elementelor din coadă (elementele cu prioritatea 0 vor fi eliminate din coadă).
- Obținerea elementului maxim din coadă (ca valoare).
- Obținerea valorii priorității maxime/minime din coadă.

6 List

Implementați clasa **List** (listă de numere intregi, reprezentată ca listă dublu înlănțuită circulară) care va avea următoarele funcționalități:

- Inserarea, ștergerea și căutarea unui element.
- Supraîncărcarea operatorului [ ] pentru accesarea elementului de pe poziția i.
- Supraîncărcarea operatorului + care să efectueze reuniunea a două liste (cu duplicate).
- Supraîncărcarea operatorilor < și > care să compare două liste (compararea se face după suma elementelor).
- Supraîncărcarea operatorului * pentru înmulțirea cu un scalar.
- Obținerea sumei elementelor din listă.
- Obținerea numărului de elemente din listă.
- Inversarea elementelor din listă.
- Determinarea elementului minim și maxim din listă.

7 Martirce rară (Sparse matrix)

Implementați clasa **SparseMatrix** (matrice în care multe elemente sunt 0, prin urmare else sunt omise, în memorie păstrându-se doar elementele nenule). Clasa va avea următoarele funcționalități:

- Supraîncărcarea operatorului + pentru adunarea a doua matrici.
- Supraîncărcarea operatorului [ ] pentru a obținerea liniei i.
- Supraîncărcarea operatorului - pentru scăderea a două matrici.
- Supraîncărcarea operatorului * pentru înmulțirea a doua matrici și înmulțirea cu un scalar.
- Obținerea numărului de linii, respectiv coloane.
- Supraîncărcarea operatorului ^ pentru ridicarea la putere a numei matrici.