Proiectul I

1 Arbori binari de căutare AVL

Implementați clasa **ArboreAVL** care sa ofere următoarele funcționalități:

- *Inserarea, ștergerea și căutarea unui element. 10p
- Parcurgere SRD(inordine), SDR(preordine), RSD(postordine). 10p
- Echilibrări ale arborilor (unde este cazul). 10p
- Supraîncărcarea operatorului + care să efectueze reuniunea a doi arbori (fără duplicate). 5p
- Supraîncărcarea operatorilor < și > care să efectueze compararea între elementele maxime a doi arbori. 4p
- Supra încărcarea operatorului - pentru difereța a doi arbori (ca diferența de mulțimi). 6p
- Obținerea numărul de elemente din arbore. 3p

2 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). 7p
- *Inserarea, ștergerea și căutarea unui element. 10p
- Păstrarea elementelor în ordine crescătoare. 5p
- Supraîncărcarea operatorului [ ] pentru accesarea elementului de pe poziția i. 3p
- Supraîncărcarea operatorului + care să efectueze reuniunea a două mulțimi (fără duplicate). 5p
- Supraîncărcarea operatorilor < și > care să compare cardinalul a două mulțimi. 4p
- Supraîncărcarea operatorului * pentru înmulțirea cu un scalar. 4p
- Obținerea sumei elementelor din mulțime. 4p
- Obținerea elementelor pare, respectiv impare din mulțime. 3p
- Obținerea numărul de elemente din mulțime. 3p

3 Graf

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

- Să se memoreze cât mai eficient structura de graf. 10p
- *Parcurgerea în lățime și lungime pornind dintr-un nod dat. 10p
- Distanța dintre două noduri date. 5p
- Supraîncărcarea operatorului [ ] care să întoarcă lista de adiacentă a unui nod sub formă de vector. 5p
- Supraîncărcarea operatorilor $<$ și $>$ care să compare numărul de noduri și numărul de muchii a două grafuri (Fie G_1 și G_2 care conțin n_1 noduri și m_1 muchii, respectiv n_2 noduri și m_2 muchii. $G_1 > G_2$ dacă $n_1 > n_2$ sau $n_1 == n_2$ && $m_1 > m_2$). 4p
- Stabilirea dacă graful este arbore. 4p
- Determinarea numărului de componente conexe din graf. 4p
- Adăugarea unei muchii la graf. 3p.
- Obținerea numărului de noduri, respectiv muchii, ale grafului. 3p

4 Matrice

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

- *Supraîncărcarea operatorului + pentru adunarea a doua matrici. 4p
- *Supraîncărcarea operatorului [ ] pentru a obținerea liniei i. 2p
- *Supraîncărcarea operatorului - pentru scăderea a două matrici. 4p
- Supraîncărcarea operatorului * pentru înmulțirea a doua matrici și înmulțirea cu un scalar. 10p
- Calcularea determinantului matricei. 20p
- Determinarea inversabilității matricei. 2p
- Obținerea unei matrici de ordin inferior prin eliminarea unei coloane, unei linii sau unei linii și unei coloane. 4p
- Obținerea numărului de linii, respectiv coloane. 2p

5 Polinoame

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

- *Calcularea valorii polinomului într-un anumit punct. 4p
- *Supraîncărcarea operatorului + pentru adunarea a două polinoame. 4p
- *Supraîncărcarea operatorului [ ] pentru obținerea termenului cu gradul i. 2p
- Supraîncărcarea operatorului * pentru înmulțirea a două polinoame și înmulțirea cu scalar. 10p
- Supraîncărcarea operatorului / pentru împărțirea a două polinoame. 20p
- Adăugarea și eliminarea unui termen de grad i. 6p
- Obținerea gradului polinomului. 2p

6 Numere Mari

Implementați clasa **BigInt** care va memora numere întregi foarte mari. Clasa va oferi următoarele funcționalități:

- *Supraîncărcarea operatorilor + și - pentru adunarea, respectiv scăderea, numerelor întregi. 10p
- Supraîncărcarea operatorilor * și / pentru înmulțirea, respectiv împărțirea, numerelor întregi. 15p
- Constructorul cu parametri și operatorul = să poată fi folosite cu numere întregi și șiruri de caractere. 7p
- Ridicarea la putere. 4p
- Determinarea parității numărului. 3p
- Determinați dacă numărul e palindrom. 5p
- Suma cifrelor numărului. 4p

7 Coadă cu Priorități

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

- *Adăugarea unui nou element în coadă. 5p
- *Eliminarea unui element dintr-o coadă. 5p
- Obținerea numărului de elemente din coadă. 5p
- Supraîncărcarea operatorului + pentru fuziunea a două cozi. 5p

- Supraîncărcarea operatorilui ++ care va crește cu 1 prioritățile elementelor din coadă. 5p
- Supraîncărcarea operatorilui – care va scădea cu 1 prioritățile elementelor din coadă (elementele cu prioritatea 0 vor fi eliminate din coadă). 7p
- Obținerea elementului maxim din coadă (ca valoare). 4p
- Obținerea valorii priorității maxime din coadă. 3p
- Gestionarea cât mai eficientă a structurii de date. 10p

8 Cerințe globale

Următoarele cerințe sunt valabile pentru toate proiectele:

- Comentarii în cod care să ofere detalii despre modul de implementare. 4p
- Indentarea adecvată a codului. 4p
- Utilizarea unei convenții de denumire a variabilelor și a metodelor. 4p
- *Separarea declarației de implementare a clasei/claselor. Declararea se va face în .h (header) și implementare în .cpp. 5p
- *Supraîncărcarea operatorilor de citire și afișare (<< și >>). 5p
- *Implementatea constructorilor cu parametri, fără parametri, de copiere și supraîncărcarea operatorului =. 10p
- *Implementarea destructorului. 5p
- *Alocarea dinamică a memoriei. 5p

9 Alte observații

- Fiecare clasă va suporta număr arbitrar de intrări (pentru cele în care se pot aelemente).
- 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 vector 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).
- Cerințele marcate cu * sunt necesare pentru a lua în considerare proiectul.
- Proiectul trebuie să compileze și să ruleze un mic demo care să demonstreze că funcționalitățile cerute au fost implementate corect.

- 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.
- Punctajul este împărțit astfel: 48p cerințe specifice, 42p cerințe globale și 10p oficiu.
- Deadline: 18-03-2018 23:59.