Tema 5 - Arbori binari de căutare

1. **Arbore binar de căutare** Implementaţi un arbore binar de căutare cu chei numere întregi. Utilizaţi o structură NOD, care are un câmp de tip int, ce stochează cheia nodului şi trei câmpuri de tip pointer la NOD pentru fiul stâng, fiul drept şi părintele nodului. De asemenea structura NOD dispune de un constructor care setează câmpul int la o valoare transmisă prin parametru şi câmpurile de tip pointer la NOD le iniţializează cu pointer nul. Utilizaţi apoi o structură de tip **SearchTree**, care are ca membru rădăcina *root* de tip pointer la NOD În plus structura trebuie să aibă metodele:

   - INSERT(int *key*) - inserează un nou nod în arbore cu cheia *key*. Dacă cheia deja există, nu se va insera. (0.25 p)
   - MAXIM(NOD *x) / MINIM(NOD *x)- returnează nodul cu cheia maximă / minimă din subarborele de rădăcină x(0.25 p ambele funcţii)
   - SUCCESOR(NOD *x) / PREDECESOR(NOD *x) - returnează nodul care este succesorul / predecesorul nodului x (0.25 p ambele funcţii)
   - FIND(int *key*) - returnează nodul cu cheia *key* dacă există sau pointer nul altfel. (0.25 p)
   - DELETE(int *key*) - şterge din arbore nodul cu cheia *key* dacă există (0.25 p)
   - ERASE(NOD *x) - şterge din arbore nodul x (care a fost mai întâi identificat prin FIND) (0.25 p)
   - PRINT_TREE(int opt) - afişază arborele în preordine (dacă opt=1), inordine (dacă opt=2), în postordine (dacă opt=3), pe niveluri (dacă opt=4). (0.5 p dintre care 0.25 pentru primele 3 afişări şi 0.25 pentru a 4-a).
   - CONSTRUCT - construieşte un AB căutare pornind de la un vector de chei. (0.25p)
   - EMPTY() - verifică dacă arborele este vid. (0.25 p)
   - CLEAR() - şterge toate nodurile din arbore (0.25 p)

Structura trebuie să dispună de un constructor care iniţializează rădăcina cu pointer nul. Funcţiile Maxim/minim/succesor/predecesor/find returnează un pointer. În funcţia *main* se declară o variabilă de tip **SearchTree** şi se foloseşte un *menu* implementat cu ajutorul unei instrucţiuni *switch*, prin care utilizatorul să poată selecta oricare dintre operaţiile de inserţie, căutare, ştergere, minim, maxim, succesor, predecesor, afişare în cele 4 moduri - la alegere. (1 p)

2. **Implementaţi un arbore roşu-negru**. Utilizaţi o structură NOD care dispune de un câmp informaţie, un câmp culoare (NU de tip string) şi câmpuri de tip pointer pentru fiii stâng şi drept şi pentru părinte. De asemenea structura NOD trebuie să dispună de un constructor care setează câmpul informaţie cu valoarea transmisă prin parametru, câmpul culoare la roşu şi câmpurile de tip pointerîn mod adecvat. Utilizaţi o structură ARN care dispune de un membru de tip pointer la NOD, numit *root* şi un câmp de tip pointer la NOD numit NIL, care este nodul santinelă. În plus dispune de funcţiile:

   - INSERT(int *key*) - inserează un nou nod în arbore cu cheia *key*. Dacă cheia deja există, nu se va insera. (0.25 p)
   - INSERT_REPARE - reface balansare după inserţie (0.75 p)
   - MAXIM(NOD *x) / MINIM(NOD *x)- returnează nodul cu cheia maximă / minimă din subarborele de rădăcină x (0.25 p ambele funcţii)
   - SUCCESOR(NOD *x) / PREDECESOR(NOD *x) - returnează nodul care este succesorul / predecesorul nodului x (0.25 p ambele funcţii)
   - FIND(int *key*) - returnează nodul cu cheia *key* dacă există sau pointer nul altfel. (0.25 p)
   - DELETE_REPARE(NOD *x) - reface balansarea arborelui după ştergere - (1p)
   - ROT_ST, ROT_DR - funcţiile de rotaţie - (0.25 p)
   - CLEAR() - şterge toate nodurile din arbore (0.25 p)
   - EMPTY() - verifică dacă arborele este vid. (0.25 p)
   - PRINT_TREE(int opt) - afişază arborele în preordine (dacă opt=1), inordine (dacă opt=2), în postordine (dacă opt=3), pe niveluri (dacă opt=4). (0.5 p dintre care 0.25 pentru primele 3 afişări şi 0.25 pentru a 4-a). Trebuie afişată şi culoarea pentru fiecare nod.
   - CONSTRUCT - construieşte un ARN pornind de la un vector de chei. (0.25p)

Structura trebuie să dispună de un constructor care iniţializează rădăcina şi santinela în modul prezentat la curs. În funcţia *main* se declară o variabilă de tip ARN şi se foloseşte un *menu* implementat cu ajutorul unei instrucţiuni *switch*, prin care utilizatorul să poată selecta oricare dintre operaţiile de inserţie, construire (cu construct), căutare, minim, maxim, afişare în cele 4 moduri - la alegere, golire (clear). (0.75 p) **Observaţie:** Această problemă se poate rezolva prin adaptarea programului de la pb. 1. Funcţiile comune vor fi punctate o singură dată.

3. **Implemetarea arborelui roşu-negru asemenea containerului map din biblioteca STL.** Modificaţi arborele implementat la punctul anterior astfel încât fiecare nod să reţină o pereche de valori: cheia cu ajutorul căreia se va determina poziţia fiecărui nod, precum şi valoarea asociată cheii.(1p)
   Arborele trebuie să suporte valori de orice tip, astfel cheia şi valoarea vor fi template.(1p)
   Implementarea operatorului paranteze patrate: [] care să returneze valoarea asociat cheii dată ca parametru, precum şi a unui iterator cu ajutorul căruia se va putea parcurge arborele in ordine SRD. (2p)

4. **Arbore de intervale: problema donaţiei**: Mihai doreşte să se doneze parte din lucrurile şi hainele pe care nu le mai foloseşte. El este însă foarte ocupat. În oraşul său sunt mai multe centre de acest tip, iar aceste centre sunt deschise doar în zilele în care există cel puţin un voluntar disponibil astfel că pentru fiecare centru se conosc zilele şi intervalele orare în care sunt deschise. Pentru Mihai nu este importantă ziua în şi nici centrul, dar el poate doar într-un anumit interval orar.

   Scrieţi un program eficient, care să îl ajute pe Mihai să găsească un o zi şi un interval orar în care se poate vaccina. (trebuie ca intervalul orar în care poate Mihai să se intersecteze cu intervalul orar propus de program).
   Atenţie: Este problemă suplimentară la ARN utilizînd îmbogăţirea structurii de date. (1,5p).

   Observaţii:

   - implementaţi pentru aceasta un arbore de intervale (având la bază un ARN).
   - se permite adăugarea de intervale oare noi, precum şi eliminarea altora (care au trecut de exemplu).
   - trebuie scrisă o funcţie, care are ca parametru intervalul orar dorit şi returnează o propunere corespunzătoare.

5. **B-secvenţă**: Numim B-secvenţă un şi de n numere $a_1, a_2, \ldots, a_n$ cu următoarele proprietăţi:

   - $a_1 < a_2 < \ldots < a_j$ şi $a_j > a_{j+1} > \ldots > a_n$
   - fiecare element, cu excepţia maximului apare de cel mult 2 ori în şir: o dată în partea crescătoare şi eventual o dată în partea descrescătoare a şirului
   - toate elementele din partea descrescătoare se regăsesc şi în partea crescătoare.

   Se citeşte o astfel de secvenţă S dintr-un fişier. Apoi se realizează K operaţii în modul următor: - pentru fiecare operaţie se citeşe o valoare val. Această valoare se inserează în secvenţa S, numai dacă după inserţie se păstrează proprietăţile definite mai sus. De asemenea după fiecare operaţie se primeşte un mesaj, care indică dacă operaţia a putut fi efectuată şi se afişează noua secvenţă. Folosiţi **set** din STL. (2p)

6. **Dicţionar**: Se citeşte dintr-un fişier un text în care cuvintele sunt separate prin spaţii. Pot exista şi semne de punctuaţie. Să se afişeze cuvintele citite în ordine alfabetică. Fiecare cuvânt se afişază o singură dată, având alături numărul de apariţii în text. Utilizaţi **map** din STL. Semnele de punctuaţie se ignoră. (1p)

7. **Lista de aşteptare**. Se consideră lista de aşteptare de la admiterea unei facultăţi. Dacă unul dintre candidaţii care au fost declaraţi admişi îşi retrage dosarul, locul acestuia va fi oferit primei persoane de pe lista de aşteptare.

   Ordinea candidaţilor de pe lista de aşteptare este influenţată de media de admire(sau a notei de la examenul de bacalaureat la caz de medii egale). Să se construiască o astfel de listă folosind **map** şi să se afişeze primii 3 candidaţi de pe această listă.Să se verifice dacă candidatul cu numele Ionescu se află pe această listă.(1p)

Observaţii:

- Funcţiile INSERT, FIND, DELETE, ERASE, CLEAR, PRINT_TREE, SUCCESOR, PREDECESOR, MINIM , MAXIM, CONSTRUCT, EMPTY, funcţiile de rotaţie etc. se punctează doar o dată, oricâte dintre probleme au fost rezolvate!
- Pentru cod copiat de pe net nota finala nu poate depăsi 4