Tema 3

Cerințe globale și obligatorii

- Alocare dinamică a memorie;
- Indentare și comentarea adecvată a codului;
- Utilizarea unei convenții de denumire a variabilelor, metodelor și claselor, cu specificarea acesteia;
- Este interzisă folosirea STL-urilor dacă, prin folosirea lor, rezolvarea devine trivială (e.g. daca tema voastra este multiset si folositi clasa multiset din STL, nota pe rezolvare va fi 0);
- Utilizarea a cat mai multe concepte POO invatate
- Utilizarea assert pentru testarea funcționalităților;
- Tema trebuie să compileze fără a utiliza anumite flag-uri de compilare (cu excepția cazurilor în care pentru compilare este necesară o anumită versiune de C++) și să respecte standardele C++ pentru sintaxă;
- Deadline: 17 mai 23:59;

1 Tabel de dispersie

`HashMap <class K, class V, class F = KeyHash<K>>`

Implementați o clasă template pentru hash map. Clasa KeyHash<K> va fi folosita pe post de functie hash default pentru calcularea hash-urilor pe chei. În caz de coliziune HashMap-ul va păstra valorile într-o listă înlănțuită. Clasa HashMap trebuie să ofere:

- constructor fără parameterii care inițializează un HashMap gol;
- constructor de copiere și operator de atribuire;
- metodă de adăugare, căutare și ștergere element nou în HashMap;
- metodă ce întoare un vector (STL) cu valorile cunoscute pentru o anumită cheie;
- metodă pentru obținerea numărului de chei distincte din HashMap;
- supraîncărcarea operatorului [] care întoarce prima valoare cunoscută pentru o cheie;
- supraîncărcarea operatorului de afișare;

2 Dicționar

`Dictionary <class K, class V, class F = KeyComp<K>>` Implementați o clasă pentru un dicționar (perechi cheie-valoare). Clasa KeyComp<K> va fi folosita pe post de comparator default pentru chei in dictionar. Clasa Dictionary trebuie să ofere:

- constructor fără parametrii care inițializează un dicționar gol;
- constructor de copiere și operatorul de atribuire;
- metodă de adăugare a unei perechi noi cheie valoare (dacă se adaugă o cheie care există deja în dicționar, se va suprascrie vechea valoare);
- metodă pentru ștergerea unei perechi din dicționar (după cheie);
- metodă de căutare după cheie;
- metodă ce elimină toate elementele din dicționar;
- supraîncărcarea operatorului [] pentru obținerea valorii salvate pentru o anumită cheie;
- supraîncărcarea operatorului de afișare;
- specializarea clasei KeyComp pentru string-uri, care sa considere doua string-uri egale daca primele n/2 caractere sunt egale, unde n este lungimea celui mai scurt string.

3 Multiset

`Multiset<class T, class F = Comparator<T>>`

Implementați o clasă template pentru multiset. Clasa oferă access rapid la elementele pe care aceasta le memorează, fără a impune o anumită restricție pe ordinea elementelor. Un element poate apărea de mai multe ori. lasa Comparator <T> va fi folosita pe post de comparator default pentru determina daca o valoare exista deja in multiset. Clasa Multiset trebuie să ofere:

- constructor fără parametrii care inițializează un multiset gol;
- constructor de copiere și operatorul de atribuire;
- metodă pentru adăugare și ștergere element din multiset (se șterge prima apariție);
- metodă care întoarce numărul de apariții ale unui element;
- metodă care verifică dacă un element se află în multiset;
- metodă care elimină toate aparițiile unui element din multiset;
- metodă care întoarce numărul de elemente distincte din multiset;
- supraîncărcarea operatorului de afișare;
- specializarea clasei Comparator pentru double, care sa considere doua valori egale daca partea zecimala este egala.

4 Set

`Set<class T, class F = Comparator<T>>`

Implementați o clasă template pentru set. Clasa oferă access rapid la elementele pe care aceasta le memorează, fără a impune o anumită restricție pe ordinea elementelor. Clasa Comparator <T> va fi folosita pe post de comparator default pentru determina daca o valoare exista deja in set. Clasa Set trebuie să ofere:

- constructor fără parametrii care inițializează un set gol;
- constructor de copiere și operatorul de atribuire;
- metodă pentru adăugare și ștergere element din set;
- metodă care verifică dacă un element se află în set;
- metodă care întoarce numărul de elemente din set;
- supraîncărcarea operatorului de afișare;
- specializarea clasei Comparator pentru int, care sa considere doi intregi egali daca au aceeasi paritate.