Proiect de programare – Cerc C, semestrul 1

Furnizarea unei biblioteci C statice pentru urmatoarele tipuri de date abstracte :

1. **Vector** (implementat ca un tablou unidimensional alocat dinamic)
   Operații
   a. CreateVector - creare instanță Vector
   b. PrintVector – afișarea elementelor
   c. AddVectorItems – adăugarea unui număr de elemente la finalul vectorului
   d. PutVectorItem – adaugă un element la indexul specificat
   e. GetVectorItem – returnează elementul de la indexul specificat
   f. DeleteVectorItem – ștergerea elementului de la indexul specificat
   g. SearchVectorItem – returnează indexul elementului care are valoarea căutată
   h. SortVector – ordonarea crescătoare a elementelor
   i. MergeVectors – combinarea a doi vectori: adăugarea la primul vector a elementelor din al doilea vector
   j. DeleteVector – ștergerea elementelor și eliberarea memoriei

2. **LinkedList** (implementat ca o listă simplu înlănțuită)
   Operații
   a. CreateLinkedList - creare instanță LinkedList
   b. PrintLinkedList – afișarea elementelor
   c. AddLinkedListItem – adăugarea unui nou element la final
   d. PutLinkedListItem – adaugă un element la pozitia specificată
   e. GetLinkedListItem – returnează elemental de la poziția specificată
   f. DeleteLinkedListItem – ștergerea și returnarea elementului cu valoarea specificată
   g. SearchLinkedListItem – returnează elementul care are valoarea căutată
   h. SortLinkedList – ordonarea crescătoare a elementelor
   i. MergeLinkedLists – combinarea a două liste: adăugarea la prima listă a elementelor din a doua listă
   j. DeleteLinkedList – ștergerea elementelor și eliberarea memoriei

3. **HashTable** (implementat folosind înlănțuire)
   Operații
   a. CreateHashTable - creare instanță HashTable
      i. Se transmite ca parametru funcția de dispersie
         1. Dacă e NULL se folosește o funcție implicită
   b. PrintHashTable – afișarea elementelor astfel încât să reflecte structura tabelei
   c. AddHashTableItem – adăugarea unui nou element
      i. Dacă factorul de umplere depășește 75% se face redimensionare (aprox. dublare)
   d. DeleteHashTableItem – ștergerea elementului cu valoarea specificată
   e. SearchHashTableItem – returnează dacă valoarea căutată este în tabela de dispersie
   f. ReHashTable – crearea unei noi tabele de hashing și redistribuirea elementelor folosind o funcție de hashing specificată, urmată de ștergerea tabelei inițiale
   g. DeleteHashTable – ștergerea elementelor și eliberarea memoriei

4. **MinHeap** (implementat ca un tablou alocat dinamic)
   Operații
   a. CreateHeap - creare instanță Heap
   b. PrintHeap – afișarea elementelor sub formă de arbore

c. AddHeapItem – adăugarea unui nou element
d. GetHeapMin – returnează elementul minim
e. DeleteHeapMin – ștergerea și returnarea elementului minim
f. DeleteHeapItem – ștergerea elementului cu valoarea specificată
g. MergeMinHeaps – combinarea a două heapuri: adăugarea la primul heap a elementelor din al doilea heap, respectând invariantul
h. DeleteHeap – ștergerea elementelor și eliberarea memoriei

5. **BinarySearchTree** (implementat ca înlănțuire de noduri)
   Operații
   a. CreateBST - creare instanță BinarySearchTree
   b. PrintBST – afișarea elementelor sub formă de arbore
   c. PreorderBST – parcurgere și afișare în preordine
   d. InorderBST – parcurgere și afișare în inordine
   e. PostorderBST – parcurgere și afișare în postordine
   f. AddBSTItem – adăugarea unui nou element la arbore
   g. SearchBSTItem – returnează elementul care are valoarea căutată
   h. DeleteBSTItem – ștergerea și returnarea elementului cu valoarea specificată
   i. MergeBSTs – combinarea a doi arbori binari de căutare: adăugarea la primul arbore a elementelor din al doilea arbore, respectând invariantul
   j. HightBST – returnează înălțimea subarborelui specificat
   k. DeleteBST – ștergerea elementelor și eliberarea memoriei

6. **BalancedBST** (implementat folosind o abordare la alegere pentru reechilibrarea arborelui – AVL, AA, B, etc.)
   Operații
   a. CreateBalancedBST - creare instanță BalancedBST
   b. PrinBalancedtBST – afișarea elementelor sub formă de arbore
   c. PreorderBalancedBST – parcurgere și afișare în preordine
   d. InorderBalancedBST – parcurgere și afișare în inordine
   e. PostorderBalancedBST – parcurgere și afișare în postordine
   f. AddBalancedBSTItem – adăugarea unui nou element la arbore
   g. SearchBalancedBSTItem – returnează elementul care are valoarea căutată
   h. DeleteBalancedBSTItem – ștergerea și returnarea elementului cu valoarea specificată
   i. MergeBalancedBSTs – combinarea a doi arbori binari de căutare echilibrați: adăugarea la primul arbore a elementelor din al doilea arbore, respectând invariantul
   j. HightBalancedBST – returnează înălțimea subarborelui specificat
   k. DeleteBalancedBST – ștergerea elementelor și eliberarea memoriei

Element opțional integrați utilizarea tipurilor generice în cadrul tipurilor abstracte de mai sus, astfel încât tipul elementelor să poată fi ales de aplicația care folosește biblioteca.

Observații:

- Se va folosi un stil de programare adecvat (comentarii, indentări, numirea variabilelor, funcțiilor, organizarea codului, etc.)
- Datele de intrare trebuie validate
- Se vor folosi doar funcții predefinite din biblioteca standard C și funcții proprii – NU funcții externe preluate din alte părți

Structura de directoare a proiectului

- **Code** (conține proiectele dezvoltate individual)
  - Biblioteca
  - Tester
- **Input** (conține fișierele de intrare – se pot partaja)
- **Output** (conține fișierele de ieșire așteptate – se pot partaja)
- **Results** (conține fișierele rezultate în urma rulării aplicației tester pe fișiere de intrare – obținute individual)

Fișierele de intrare se denumesc astfel: 001.in, 002.in, ...

Fișierele de ieșire se denumesc astfel: 001.out, 002.out, ...

Fișierele rezultate se denumesc astfel: 001.res, 002.res, ...

Observații:

- Numele instanței de tip abstract este o singură literă mare (majusculă) și apare pe aceeași linie cu comanda
- testul cu numărul 001 se consideră trecut dacă rezultatul obținut în 001.res este identic cu rezultatul așteptat, furnizat în 001.out (chiar și în cazuri de eroare)
- fișierul de intrare poate conține date invalide
- toate fișierele de input și output trebuie să respecte convenția stabilită pentru format
- valorile elementelor pe care se vor testa funcționalitățile vor începe pe 4 octeți
  - dacă integrați tipuri generice și doriți să testați și pe alte valori aplicația – vă creați individual fișierele de intrare care să corespundă diferitelor tipuri suportate
  - componentele structurilor se dau pe aceeași linie despărțite prin ; (punct și virgulă)
- numărul maxim de elemente nu va depăși 10 000 000
- biblioteca trebuie să fie independentă de aplicația Tester și de formatul fișierelor de input/output. Testerul poate interpreta comenzile și formatul fișierelor de intrare și poate scrie în fișierele de ieșire.

Recomandare: urmăriți să creați fișiere de intrare cu scenarii de utilizare cât mai variate, inclusiv cu input invalid.

Exemplu:

`Tester.exe nr` - ruleaza testul cu numarul *nr (nr.in)*
`Tester.exe nr1 nr2` - ruleaza testele cuprinse intre *nr1* si *nr2 (nr1.in – nr2.in)*
`Tester.exe runall` - ruleaza toate testele (toate fișierele de intrare din folderul Input)

Exemplu de fisier input/output:

001.in:	***001.out***:
CreateVector A AddVectorItems A 5 1 3 7 6 9 PrintVector A PutVectorItem A 2 10 PrintVector A	1 3 7 6 9 1 3 10 7 6 9

Exemple de mesaje de eroare si cazuri speciale:

Nr	Mesaj	Semnificație
1	Error: Illegal operation. Data structure does not exist	La efectuarea unei operații pe o structura inexistenta. Ex. Inserare intr-un vector care nu este creat, sau a fost deja sters
2	Error: Unrecognized command: *command*	La citirea unei comenzi necunoscute de catre Tester. Se afișează linia care nu poate fi interpretată (*command*).
3	Error: Type mismatch	Eroare la atribuirea datelor preluate din fișier la elementele structurii de date din cauza incompatibilităților de tip.
4	Error: Missing value	Se așteaptă o valoare care nu este furnizată.
5	Error: Index out of bounds	La accesul la un element de la un index în afara limitelor existente (prea mare sau prea mic).
6	Error: Unrecognized hash function	Funcția de dispersie specificată nu este cunoscută.
7	Error: Memory allocation failed	Eșuarea alocării dinamice a memoriei.
8	Cannot delete: Item not found	Elementul nu a fost regăsit în structura de date și nu poate fi șters.
9	Printing: Structure is empty	Afișarea unei structuri care nu este populată (nu conține nici un element).
10	Error: Instance unknown	Instanta de structura nu este specificată
11		