Lab 7: Binary Search Tree

Implement in C++ the given **container** (ADT) using a given representation and a **binary search tree (BST)** as a data structure. You are not allowed to use any container or data structure from STL or from any other library.

Do not implement a separate class for the binary search tree, implement the container directly!

1. **ADT Matrix** – represented as a sparse matrix where <line, column, value> triples (value ≠ 0) are memorized. The elements are stored in a BST with linked representation with dynamic allocation.
2. **ADT Matrix** – represented as a sparse matrix where <line, column, value> triples (value ≠ 0) are memorized. The elements are stored in a BST with linked representation on an array.
3. **ADT SortedBag** – using a BST with linked representation with dynamic allocation. If an element appears multiple times, it will be stored multiple times in the BST.
4. **ADT SortedBag** – using a BST with linked representation on an array. If an element appears multiple times, it will be stored multiple times in the BST.
5. **ADT SortedBag** – using a BST with linked representation with dynamic allocation. In the BST (unique element, frequency) pairs are stored.
6. **ADT SortedBag** – using a BST with linked representation on an array. In the BST (unique element, frequency) pairs are stored.
7. **ADT SortedSet** – using a BST with linked representation with dynamic allocation.
8. **ADT SortedSet** – using a BST with linked representation on an array.
9. **ADT Sorted Map** – using a BST with linked representation with dynamic allocation.
10. **ADT Sorted Map** – using a BST with linked representation on an array.
11. **ADT SortedMultiMap** – using a BST with linked representation with dynamic allocation. In the BST (key, value) pairs are stored. If a key has multiple values, it appears in multiple pairs.
12. **ADT SortedMultiMap** – using a BST with linked representation on an array. In the BST (key, value) pairs are stored. If a key has multiple values, it appears in multiple pairs.
13. **ADT SortedMultiMap** – using a BST with linked representation with dynamic allocation. In the BST unique keys are stored with a dynamic array of the associated values.
14. **ADT SortedMultiMap** – using a BST with linked representation on an array. In the BST unique keys are stored with a dynamic array of the associated values.
15. **ADT SortedList** – using a BST with linked representation with dynamic allocation. Every node of the BST will retain the number of elements to the left of the node as well.
16. **ADT SortedList** – using a BST with linked representation on an array. Every node of the BST will retain the number of elements to the left of the node as well.