L3: Linked List on Array

Implement in C++ the given **container** (ADT) using a given representation and a **linked list on array** as a data structure. You are not allowed to use the *vector* or *list* from STL or from any other library (except as a return type for the *search* operation for a SortedMultiMap, as it appears in its interface).

1. **ADT Matrix** – represented as a sparse matrix, using a SLLA with <line, column, value> triples (value ≠ 0), ordered lexicographically considering the line and column of every element.
2. **ADT Matrix** – represented as a sparse matrix, using a DLLA with <line, column, value> triples (value ≠ 0), ordered lexicographically considering the line and column of every element.
3. **ADT Bag** – using a SLLA with (element, frequency) pairs.
4. **ADT Bag** – using a DLLA with (element, frequency) pairs.
5. **ADT SortedBag** – using a SLLA with (element, frequency) pairs. Pairs are ordered based on a relation between the elements.
6. **ADT SortedBag** – using a DLLA with (element, frequency) pairs. Pairs are ordered based on a relation between the elements.
7. **ADT SortedSet** – using a SLLA where elements are ordered based on a relation between the elements.
8. **ADT SortedSet** – using a DLLA where elements are ordered based on a relation between the elements.
9. **ADT Set** – using a SLLA
10. **ADT Set** – using a DLLA
11. **ADT Map** – using a SLLA with (key, value) pairs
12. **ADT Map** – using a DLLA with (key, value) pairs
13. **ADT MultiMap** – using a SLLA with (key, value) pairs. A key can appear in multiple pairs. Pairs do not have to be ordered.
14. **ADT MultiMap** – using a DLLA with (key, value) pairs. A key can appear in multiple pairs. Pairs do not have to be ordered.
15. **ADT MultiMap** – using a SLLA with *unique* keys. Every key will be associated with a SLLA of the values belonging to that key.
16. **ADT MultiMap** – using a DLLA with *unique* keys. Every key will be associated with a DLLA of the values belonging to that key.
17. **ADT SortedMap** – using a SLLA with (key, value) pairs ordered based on a relation on the keys.
18. **ADT SortedMap** – using a DLLA with (key, value) pairs ordered based on a relation on the keys.
19. **ADT SortedMultiMap** – using a SLLA with *unique* keys ordered based on a relation on the keys. Every key will be associated with a SLLA of the values belonging to that key.
20. **ADT SortedMultiMap** – using a DLLA with *unique* keys ordered based on a relation on the keys. Every key will be associated with a DLLA of the values belonging to that key.

21. **ADT SortedMultiMap** – using a SLLA with (key, value) pairs ordered based on a relation on the keys. A key can appear in multiple pairs.
22. **ADT SortedMultiMap** – using a DLLA with (key, value) pairs ordered based on a relation on the keys. A key can appear in multiple pairs.
23. **ADT List** (interface with **TPozition = Integer, IndexedList**) – using a SLLA
24. **ADT List** (interface with **TPozition = Iterator, IteratedList**) – using a SLLA
25. **ADT List** (interface with **TPozition = Integer, IndexedList**) – using a DLLA
26. **ADT List** (interface with **TPozition = Iterator, IteratedList**) – using a DLLA
27. **ADT SortedList** (interface with **TPozition = Integer, SortedIndexedList**) – using a SLLA where elements are ordered based on a relation.
28. **ADT SortedList** (interface with **TPozition = Iterator, SortedIteratedList**) – using a SLLA where elements are ordered based on a relation.
29. **ADT SortedList** (interface with **TPozition = Integer, SortedIndexedList**) – using a DLLA where elements are ordered based on a relation.
30. **ADT SortedList** (interface with **TPozition = Iterator, SortedIteratedList**) – using a DLLA where elements are ordered based on a relation.
31. **ADT Priority Queue** – using a SLLA with (element, priority) pairs ordered based on a relation between the priorities.
32. **ADT Priority Queue** – using a DLLA with (element, priority) pairs ordered based on a relation between the priorities.