# Sorted List – Linked Section Exercises

1. Write a class based on class SortedType as an unbounded array-based implementation. If the dynamically allocated array is full, create an array double the size and move the elements into it.

2. Write a class based on class SortedType as a bounded linked implementation. Provide a parameterized constructor that takes the maximum number of items as a parameter. If function PutItem is called when the list if full, throw an exception.

3. A Sorted List ADT is to be extended by the addition of a member function Head, which has the following precondition and postcondition:

| | |
|---|---|
| Precondition | list has been initialized and is not empty. |
| Postcondition | return value is the last item inserted in the list. |

4. A Sorted List ADT is to be extended by the addition of function SplitLists, which has the following specifications:
   ***SplitLists(SortedType list, ItemType item, SortedType& list1, SortedType& list2)***

| | |
|---|---|
| Function | Divides list into two lists according to the key of item. |
| Preconditions | list has been initialized and is not empty. |
| Postconditions | - list1 contains all the items of list whose keys are less than or equal to item's key;<br>- list2 contains all the items of list whose keys are greater than item's key. |

   - Write the function definition, using an linked-based implementation.
5. The specifications for the Sorted List ADT state that the item to be deleted is in the list.
   - Rewrite the specification & implementation for DeleteItem so that the list is unchanged if the item to be deleted is not in the list using an linked-based implementation.
   - Rewrite the specification & implementation for DeleteItem so that all copies of the item to be deleted are removed if they exist using an linked-based implementation.