2025-09-10

# Lists (Solutions)

## Multiple Choices

1. Put a checkmark in the box corresponding to true statements about the List abstract data type.
	* A list contains an finite collection of elements, in a particular order.
	* A list cannot contain multiple elements with the same value.
	* A list must have a fixed number of elements.
	* A list is generally endowed with an operation to test for emptiness.
	* Only the element at the beginning of a list can be removed.
* Comments on the solution
	+ A list cannot contain an *infinite* collection of elements.
	+ A list can have repetition, the same value can be present multiple times.
	+ At any given time, a list has a fixed size.
	+ This is *generally* the case in definitions of lists.
	+ This restriction applies to queues or stacks (depending how “beginning” is interpreted), but not to lists.

## Exercises

1. Given the usual implementation of Cell and CList:
* public class CList<T>{
 private Cell first;
 private class Cell{
 public T Data { get; set; }
 public Cell Next { get; set; }
 public Cell(T dataP, Cell nextP){Data = dataP; Next = nextP;}
 }
 public CList(){first = null;}
}
* Write…
	1. … a IsEmpty property that is true if the CList calling object is empty.
	+ Solution
	+ Note that the question asks for a *property*:
	+ public bool IsEmpty{
	 get{ return first == null; }
	 }
	1. … the AddF method that add a cell at the beginning of the CList (to the left).
	+ Solution
	+ The key is to use the given Cell constructor to create the new element:
	+ public void AddF(T dataP){
	 first = new Cell(dataP, first);
	 }
	1. … a series of statements, to be inserted in a Main method, that a. create a CList object capable of containing char, b. insert the elements 'b' and '/' in it, c. displays whether it is empty using IsEmpty.
	+ Solution
	+ Remembering that IsEmpty is a property, we obtain:
	+ CList<char> myList1 = new CList<char>();
	 myList1.AddF('b');
	 myList1.AddF('/');
	 Console.WriteLine("myList1 is empty:" + myList1.IsEmpty);
1. Briefly explain the purpose of the IsReadonly property from the ICollection<T> interface, and list at least two methods in a List implementation realizing ICollection<T> that should use it.
* Solution
* This property indicates whether the ICollection<T> is read-only: if set to true, the ICollection<T> object should not accept addition or removal of elements. Hence, any method involving adding (AddF, AddL, …) or removing (Clear, RemoveF, RemoveL, RemoveI, …) values should test whether IsReadonly is true before proceeding.
1. Explain the main differences between singly linked list and doubly linked list, and name a few methods that need to be implemented differently.
* Solution
* Doubly linked lists use a Cell class that contains *two* references: in addition to containing a reference to the Cell coming “after” themselves, as in singly linked lists, they also contain a reference to the Cell that is “before” them. This also requires to manipulate two references for the list: in addition to one reference to the first element (now called Head), as in singly linked list, they contain a reference to the “last” element (called Tail).
* Clearing the list, adding and removing an element need to be implemented differently, as more references need to be updated.
1. For what operation(s) does doubly linked list provide a complexity gain over singly linked list?
* Solution
* Inserting at the end of the list is $O\left(c\right)$ for doubly linked list, but $O\left(n\right)$ for singly linked list. In general, traversing the list in reverse order is less costly if the list is doubly linked.