Recursion on Lists

* split the problem into two parts:
	+ a part that is easy to solve; and
	+ a part that is a smaller version of the main problem

We can still prove that recursions give the right answer

* prove base case is correct
* prove that the recursive case is correct, assuming the recursive call returns the right answer

e.g. minList():

* Stopping case / Base case: If the size of the list is 1, then minList() returns the one element of the list. This is correct.
* Recursive case: If the size of the list is n, then minList() returns either the first element of the list, or the minimum of the rest of the elements (more precisely, minList() called on the sublist from index 1 to index size()-1, both inclusive). Assuming the minList() call gives the right answer, then either the minimum is the first element of the list, or the minimum from the rest (returned by minList()). The minimum between those two is the minimum element in the whole list. The recursive call is correct.
* Therefore, the entire recursion returns the right answer.

Does the method halt? We need:

* size defined on nonnegative integers
* each recursive call decreases the size
* Here the size is actually the size of the list passed to minList()
* Does this decrease? In the recursive call, we call minList(sublist)
* Sublist consists of the elements of the original list from 1 to size()-1 inclusive
* Therefore the size of the recursive call size()-1 (where size() represents the size of the original list)
* Size decreases with each recursive call