CS173: Intermediate Programming

List ADT

Overview
The List class is an ordered collection of items. The items in the list must all be of the same datatype which is specified by a template. The number of items in the list is specified by the length() of the list. Items in the list are indexed from 0 (first item) to length-1 (last item). The list grows dynamically so that it is never full. Items may be inserted into the list and removed from the list. Values at individual indices may be accessed and/or changed. If a method is attempted with an invalid index (removing, adding, indexing), an error message is printed and the program terminates.

Constructors	
default	A new empty List is created. `List l;`
copy	Create a new List from an existing one. `List l1(l2);`

Access	
isEmpty	Returns true if the list is empty, false otherwise. `if (l.isEmpty()) ...`
length	Returns the number of items in the list. `for ( int i = 0; i < l.length(); i++) ...`
operator[]	Accesses (by reference) the item at the specified index. `int i = l[5]; l[3] = 10;` A program-termination error is issued if the index is invalid. Invalid indices are less than 0 or greater than length-1.
cout <<	Overload the cout << operator to print the list. `cout << l1 << endl;` Formating example: `[ 1, 2, 3 ]`

Modifiers	
append	Appends a new item onto the back of the list. `l1.append(5);`

insert(item,pos)	Inserts a new value at the specified position. Valid indices for the position are 0 ... length. Inserting at position=length is like an append. Existing values in the list are moved up one index location to make room for the new item.
	`l1.insert(5,1); // inserts 5 at index 1`
	A run-time error is generated for an invalid index.

remove	Removes an item at the specified location. Valid locations are 0...length-1; `l1.remove(3);` A run-time error is generated for an invalid index.
operator+	Concatenates two lists into a new list. Does not change either existing list. `l3 = l1 + l2;`
operator=	Assignment operator. `l1 = l2;`
clear	Removes all items from the list. `l1.clear();`
destructor	Cleans up the memory of the list.