# 09. C++11 Advanced Features - Homework Exercises

Write C++ code for solving the tasks on the following pages.

Code should compile under the C++11 standard.

You can organize your code in .h and .cpp files (or only a single .cpp file, or a single .cpp file including only .h files), but you should ONLY submit them (don’t submit compiled files, like .exe, .o, .obj, etc.) and the files for each task should be in a separate folder (e.g. a folder with files for Task 1, another for Task 2, etc.). You can include the project file(s) from the IDE you use, if you want (E.g. the .cbp file if you use Code::Blocks).

Each task in this homework expands the functionality of the same List class, but you should organize your homework so that the code in each folder should only contain the functionality available up to that task (for example, the code for Task 2 should have a List, which uses templates, supports the range-based for loop and initializer-list initialization, but should NOT have a move constructor, because that is added in Task 3)

The folder for each task should be named with the task number followed by what you feel describes the exercise in a few words.

E.g. a good name for the folder for task 2 of this homework would be:  
2.List-with-List-Initializer

Task 5 is a bonus task and is NOT included in the score for this homework.

Don’t worry about the name too much, just make sure the number is correct and that the files in the folder can be compiled when organized into a project.

## Task 1

Implement a doubly-linked List class (like the one for JA2-Task-1-List).

* Make it work for any data type, i.e. make it a class template which has a template argument for the stored data type.
* Make it support the C++11 range-based for loop*. Hint: we did something very similar to this in Exercises on OOP and Code Organization*.

For example, this code should compile and execute correctly:

|  |
| --- |
| List<int> numbers;  numbers.add(1);  numbers.add(-2);  numbers.add(42);  for (int n : numbers) {  std::cout << n << " ";  }  std::cout << std::endl;  List<std::string> words;  words.add("Guardians");  words.add("galaxy");  for (auto word : words) {  std::cout << word << " ";  }  std::cout << std::endl; |

The code above should print out the following lines:

|  |
| --- |
| 1 -2 42  Guardians galaxy |

## Task 2

Add C++11 list-initialization of elements to the List class you implemented in Task 1. The following code should compile and execute correctly, and it should have the same result as the code in Task 1.

|  |
| --- |
| List<int> numbers {1, -2, 42};  for (int n : numbers) {  std::cout << n << " ";  }  std::cout << std::endl;  List<std::string> words {"Guardians", "galaxy"};  for (auto word : words) {  std::cout << word << " ";  }  std::cout << std::endl; |

## Task 3

Implement move semantics (Move Constructor and Move Assignment Operator) for the List. Check to see that they are called correctly when initializing or assigning a List with rvalues and make sure that you have no memory leaks.

## Task 4

Use Smart Pointers to store the Nodes of the List from Task 1 and Task 2, instead of normal pointers. Ensure that there are no memory leaks. Hint: be careful – you will need different types of smart pointers – you don’t want circular shared\_ptr, because those will never free their owned memory.

## Task 5\*

Bonus Task – this task is not mandatory and will not count towards your homework score. You can submit your homework without it.

Allow your List class to be inserted in an unordered\_set.

* Two Lists are considered equal if and only if they have the same length and have the same elements in the same order.
* The hash code of a List is calculated based on the sum of the hash codes of each of its elements, but before adding an item’s hash code to the sum, it is multiplied by its position (starting from 1) – for example, the List {13, 42, 69} will have the hash code: std::hash(13) \* 1 + std::hash(42) \* 2 + std::hash(69) \* 3.