**Modern & Effective C++: Домашня робота**

Напишіть власну реалізацію однозв’язаного списку (ви можете переглянути **std::forward\_list** (<https://en.cppreference.com/w/cpp/container/forward_list>) як довідник щодо того, що потрібно реалізувати), у якому визначено такі операції:

1. Основні операції зі списком: конструктор, вставка елементів на початку списку, ітерація по списку, розмір списку, видалення елементів.
2. Move-операції (move-конструктор, move-присвоєння)
3. Метод для вставки елементів на початку повинен мати перевантаження для посилання rvalue і правильно обробляти move.
4. Створення списку за допомогою конструктора, який приймає std::initializer\_list.
5. Список повинен бути шаблонним і працювати не тільки з примітивними значеннями, а й з об'єктами (і правильно звільняти пам'ять)
6. В ідеалі ви повинні прагнути до інтерфейсу класу на кшталт std::forward\_list, але вам не потрібно реалізовувати абсолютно всі методи - кастомний конструктор і методи emplace\_\* не обов'язкові (але це не буде мінусом). Сортування, merge, та unique робити не потрібно.
7. БОНУСНІ БАЛИ: реалізуйте метод split\_when, який дозволяє вам розділити список на два, що визначає позицію, з якої елемент розділити, на основі логічного значення, що повертається переданою **функцією предикату** (<https://stackoverflow.com/questions/3230944/what-does-predicate-mean-in-the-context-of-computer-science>). Сам предикат можна передавати за допомогою std::function, наприклад:

// Predicate type

**template**<**typename** T>

**using** SplitPredicate = std::function<**bool**(**const** T& value)>;

// Proposed function signature

**template**<**typename** T>

**class** **LinkedList**

{

// ... Other methods ...

LinkedList<T> split\_when(SplitPredicate<T> condition);

// ... Other methods ...

};

// Expected usage

LinkedList<**int**> original = { **1**, **42**, **3** };

**auto** tail = original.split\_when(

[](**const** **int**& value)){ **return** value == **42**; }

);

// original now is { 1 }

// tail now is { 42, 3 }

Додатково:

* Витоки пам'яті/неправильне використання пам'яті також оцінюються; якщо std::unique\_ptr можна буде використовувати, то це буде плюсом.
* Обов'язково перевірте свою роботу на компіляцію/працездатність перед подачею.
* Для розробки рекомендується використовувати Visual Studio 2019. Але якщо дуже хочеться, а виходу зовсім немає, можна використовувати що завгодно (але вказати версію компілятора та ОС!!!)