Custom list Implementation
Homework

Part 1.
Custom list implementation

1. Implement custom list class:
   ```
   template <typename T>
   class _list
   ```
   You can use the one that we implemented in the last 2 classes and extend it for these homework problems, but it's preferable to implement it from scratch (with a fixed version of tail, so that for example `cout << *(--l.end());` will print the last element). All the 12 test functions should be passed correctly on your list implementation.
2. Implement a **`reverse`** function (two versions of it: **`reverse_iterative`** and **`reverse_recursive`**). **`test13`** and **`test14`** should pass on your implementation.
3. Implement **`push_front`** and **`pop_front`** functions. **`test14`**.
4. Choose some interesting methods from std::list and implement them (maybe a constructor with initializer_list parameter or merge method, etc.).

Part 2.
Algorithmic Problems on our list implementation

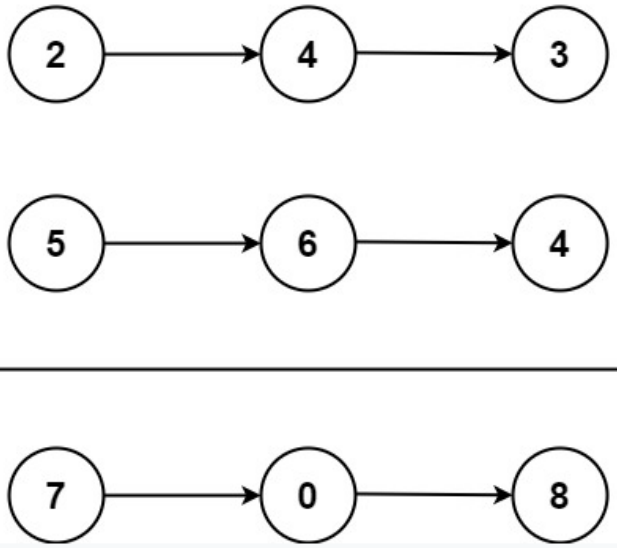
All the three problems you should implement in the **`list_problems.h`** file in the corresponding functions.

Problem 1.

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:




```
Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [7,0,8]
Explanation: 342 + 465 = 807.
```

Example 2:

```
Input: l1 = [0], l2 = [0]
Output: [0]
```

Example 3:

```
Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
Output: [8,9,9,9,0,0,0,1]
```

You should implement the solution inside **`problem1_solution::addTwoNumbers`** function. There are three tests testing your implementation. **`problem1_test1(), problem1_test2(), problem1_test3();`**

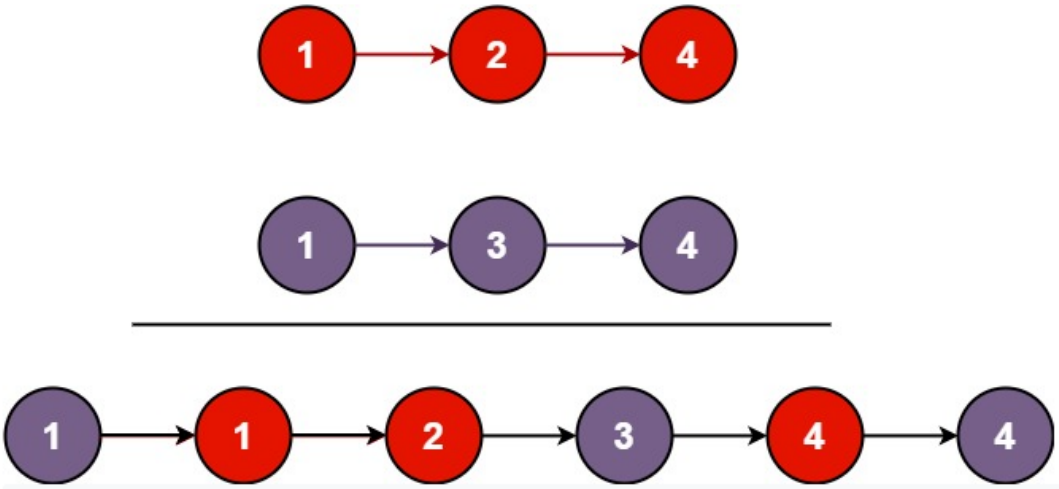
Problem 2.

You are given the two sorted linked lists _list<int>& l1, const _list<int>& l2.

Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the new merged list.

Example 1:




```
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
```

Example 2:

```
Input: list1 = [], list2 = []
Output: []
```

Example 3:

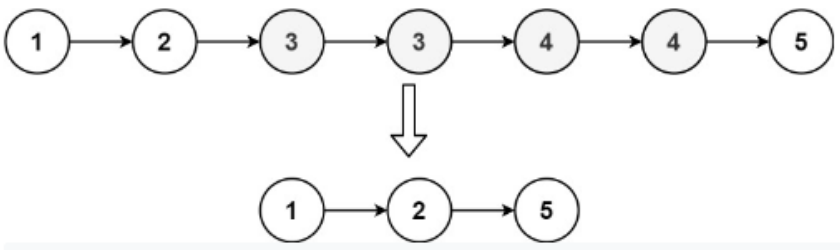
```
Input: list1 = [], list2 = [0]
Output: [0]
```

You should implement the solution inside **`problem2_solution::mergeTwoLists`** function. There are three tests testing your implementation. **`problem2_test1(), problem2_test2(), problem2_test3();`**

Problem 3.

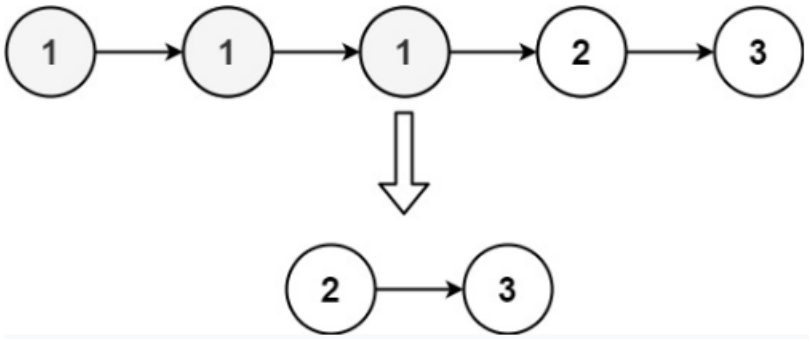
delete all nodes from `_list<T>` that have duplicate numbers, leaving only distinct numbers from the original list.

Example 1:




```
Input: head = [1,2,3,3,4,4,5]
Output: [1,2,5]
```

Example 2:




```
Input: head = [1,1,1,2,3]
Output: [2,3]
```

You should implement the solution inside **`template <typename T> void _list<T>::deleteDuplicates`** function. There are three tests testing your implementation. **`problem3_test1(), problem3_test2(), problem3_test3();`**

You also have another homework: that is the implementation of LRU replication data structure (another PDF). If you solve all of the problems you can consider solving the following leetcode problems as extra homework.

swap-nodes-in-pairs
reverse-nodes-in-k-group
linked-list-cycle-ii
reorder-list