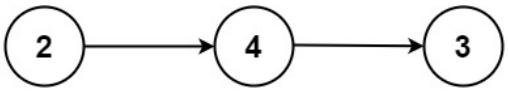
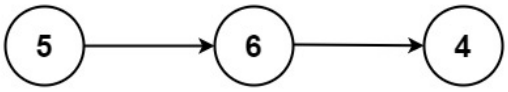
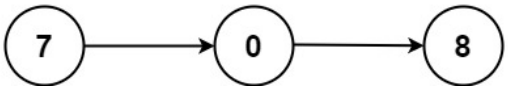
Wednesday 1/26/22

Linked-lists: https://www.youtube.com/watch?v=njTh_OwMljA
Hashtables: https://www.youtube.com/watch?v=shs0KM3wKv8&t=238s

HW1:

1	Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to the target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order. Example 1: Input: nums = [2,7,11,15], target = 9 Output: [0,1] Explanation: Because nums[0] + nums[1] == 9, we return [0, 1]. Example 2: Input: nums = [3,2,4], target = 6 Output: [1,2] Example 3: Input: nums = [3,3], target = 6 Output: [0,1] Constraints: 2 <= nums.length <= 104 -109 <= nums[i] <= 109 -109 <= target <= 109 Only one valid answer exists. Follow-up: Can you come up with an algorithm that is less than O(n2) time complexity? class Solution { public: vector<int> twoSum(vector<int>& nums, int target) { } };

2	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] Constraints: The number of nodes in each linked list is in the range [1, 100]. 0 <= Node.val <= 9 It is guaranteed that the list represents a number that does not have leading zeros. /**

	`` * Definition for singly-linked list. * struct ListNode { * int val; * ListNode *next; * ListNode() : val(0), next(nullptr) {} * ListNode(int x) : val(x), next(nullptr) {} * ListNode(int x, ListNode *next) : val(x), next(next) {} * }; */ class Solution { public: ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) { } }; ```
3	You are given an array of k linked-lists lists, each linked-list is sorted in ascending order. Merge all the linked-lists into one sorted linked-list and return it. Example 1: Input: lists = [[1,4,5],[1,3,4],[2,6]] Output: [1,1,2,3,4,4,5,6] Explanation: The linked-lists are: [1->4->5, 1->3->4, 2->6] merging them into one sorted list: 1->1->2->3->4->4->5->6 Example 2: Input: lists = [] Output: [] Example 3: Input: lists = [[]] Output: [] Constraints:

k == lists.length
0 <= k <= 10^4
0 <= lists[i].length <= 500
-10^4 <= lists[i][j] <= 10^4
lists[i] is sorted in ascending order.
The sum of lists[i].length won't exceed 10^4.

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {

    }
};
```