1. Reverse Integer (Leetcode:- 7)

Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

Input: x = 123
Output: 321
Example 2:

Input: x = -123
Output: -321
Example 3:

Input: x = 120
Output: 21

Constraints:

-231 <= x <= 231 - 1

2. Two Sum (Leetcode:- 1)

Given an array of integer numbers and an integer target, return indices of the two numbers such that they add up to 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.

3. Search Insert Position (LeetCode-35)

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [1,3,5,6], target = 5
Output: 2
Example 2:

Input: nums = [1,3,5,6], target = 2
Output: 1
Example 3:

Input: nums = [1,3,5,6], target = 7
Output: 4

Constraints:

1 <= nums.length <= 104
-104 <= nums[i] <= 104
nums contains distinct values sorted in ascending order.
-104 <= target <= 104