1. Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where:

- '?' Matches any single character.
- '*' Matches any sequence of characters (including the empty sequence).

The matching should cover the entire input string (not partial).

Example 1:

> Input: s = "aa", p = "a"
> Output: false
> Explanation: "a" does not match the entire string "aa".

Example 2:

> Input: s = "aa", p = "*"
> Output: true
> Explanation: '*' matches any sequence.

Example 3:

> Input: s = "cb", p = "?a"
> Output: false
> Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.

Example 4:

> Input: s = "adceb", p = "*a*b"
> Output: true
> Explanation: The first '*' matches the empty sequence, while the second '*' matches the substring "dce".

Example 5:

> Input: s = "acdcb", p = "a*c?b"
> Output: false

Constraints:

- 0 <= s.length, p.length <= 2000
- s contains only lowercase English letters.
- p contains only lowercase English letters, '?' or '*'.

2. 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.

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

Example 4:

Input: nums = [1,3,5,6], target = 0
Output: 0

Example 5:

Input: nums = [1], target = 0
Output: 0

Constraints:

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

3.