LeetCode 167: Two Sum II - Input Array Is Sorted

Problem:
Given a 1-indexed array of integers numbers that is already sorted in non-decreasing order, find two numbers such that they add up to a specific target number. Return the indices of the two numbers (1-indexed). **Example:**
Input: numbers = [2,7,11,15], target = 9
Output: [1,2]
Explanation: The sum of 2 and 7 is 9. Therefore, index1 = 1, index2 = 2.

Approach 1: Brute Force (O(n²))

```
// C++ O(n^2) Brute Force
#include <bits/stdc++.h>
using namespace std;

vector<int> twoSum(vector<int>& numbers, int target) {
    int n = numbers.size();
    for(int i = 0; i < n; i++) {
        for(int j = i + 1; j < n; j++) {
            if(numbers[i] + numbers[j] == target)
                return {i + 1, j + 1};
        }
    }
    return {};
}

int main() {
    vector<int> nums = {2,7,11,15};
    int target = 9;
    vector<int> res = twoSum(nums, target);
    cout << "[" << res[0] << "," << res[1] << "]" << endl;
    return 0;
}

// Java O(n^2) Brute Force
class Solution {
    public int[] twoSum(int[] numbers, int target) {
        for (int i = 0; i < numbers.length; i++) {
            for (int j = i + 1; j < numbers.length; j++) {
                if (numbers[i] + numbers[j] == target)
                    return new int[]{i + 1, j + 1};
            }
        }
        return new int[]{};
    }

    public static void main(String[] args) {
        Solution s = new Solution();
        int[] res = s.twoSum(new int[]{2,7,11,15}, 9);
        System.out.println("[" + res[0] + "," + res[1] + "]");
    }
}

# Python O(n^2) Brute Force
def twoSum(numbers, target):
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if numbers[i] + numbers[j] == target:
                return [i + 1, j + 1]

print(twoSum([2,7,11,15], 9))  # Output: [1, 2]
```

Approach 2: Two Pointer (O(n))

```cpp
// C++ Two Pointer O(n)
#include <bits/stdc++.h>
using namespace std;

vector<int> twoSum(vector<int>& numbers, int target) {
    int left = 0, right = numbers.size() - 1;
    while (left < right) {
        int sum = numbers[left] + numbers[right];
        if (sum == target)
            return {left + 1, right + 1};
        else if (sum < target)
            left++;
        else
            right--;
    }
    return {};
}

int main() {
    vector<int> nums = {2,7,11,15};
    int target = 9;
    vector<int> res = twoSum(nums, target);
    cout << "[" << res[0] << "," << res[1] << "]" << endl;
    return 0;
}
```

```java
// Java Two Pointer O(n)
class Solution {
    public int[] twoSum(int[] numbers, int target) {
        int left = 0, right = numbers.length - 1;
        while (left < right) {
            int sum = numbers[left] + numbers[right];
            if (sum == target)
                return new int[]{left + 1, right + 1};
            else if (sum < target)
                left++;
            else
                right--;
        }
        return new int[]{};
    }

    public static void main(String[] args) {
        Solution s = new Solution();
        int[] res = s.twoSum(new int[]{2,7,11,15}, 9);
        System.out.println("[" + res[0] + "," + res[1] + "]");
    }
}
```

```python
# Python Two Pointer O(n)
def twoSum(numbers, target):
    left, right = 0, len(numbers) - 1
    while left < right:
        total = numbers[left] + numbers[right]
        if total == target:
            return [left + 1, right + 1]
        elif total < target:
            left += 1
        else:
            right -= 1

print(twoSum([2,7,11,15], 9))  # Output: [1, 2]
```