<https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/>

**Two Sum II - Input Array Is Sorted**

**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. Let these two numbers be numbers[index1] and numbers[index2] where 1 <= index1 < index2 <= numbers.length.**

**Return the indices of the two numbers, index1 and index2, added by one as an integer array [index1, index2] of length 2.**

**The tests are generated such that there is exactly one solution. You may not use the same element twice.**

**Your solution must use only constant extra space.**

Example 1:

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. We return [1, 2].

Example 2:

Input: numbers = [2,3,4], target = 6

Output: [1,3]

Explanation: The sum of 2 and 4 is 6. Therefore index1 = 1, index2 = 3. We return [1, 3].

Example 3:

Input: numbers = [-1,0], target = -1

Output: [1,2]

Explanation: The sum of -1 and 0 is -1. Therefore index1 = 1, index2 = 2. We return [1, 2].

Constraints:

2 <= numbers.length <= 3 \* 104

-1000 <= numbers[i] <= 1000

numbers is sorted in non-decreasing order.

-1000 <= target <= 1000

The tests are generated such that there is exactly one solution.

**Method 1: (Brute Force)**

For each element find key = target – element.

Time Complexity: O(n2)

Space Complexity: O(1)

**Method 2: (Two pointer)**

Use two pointers i and j to point to start and end index.

Using two pointers we can get every possible pair combination in the array.

If sum of numbers at index i and j is less than target decrement j

If it is greater than target increment I, otherwise return indices.

Time Complexity: O(n) *[]*

Space Complexity: O(1) *[]*

vector<int> twoSum(vector<int>& numbers, int target) {

        int i=0, j=numbers.size()-1;

        int x;

        vector<int> ans(2);

        while(i<j){

            x=numbers[i] + numbers[j];

            if(x==target){

                ans[0]=i+1;

                ans[1]=j+1;

                break;

            }

            else if(x>target)

                j--;

            else i++;

        }

        return ans;

    }