TWO POINTERS TECHNIQUE

(1) VALID PALINDROME

```
def isPalindrome(s: str) -> bool:
  left, right = 0, len(s) - 1
  while left < right:
      if not s[left].isalnum():
          left += 1
      elif not s[right].isalnum():
          right -= 1
      elif s[left].lower() != s[right].lower():
          return False
      else:
          left, right = left + 1, right - 1
  return True

print(isPalindrome("Was it a car or a cat I saw?"))
print(isPalindrome("@#racecar%^"))
```

(2) CONTAINER WITH MOST WATER

```
def max_area_two_pointers(heights):
  max_area = 0
  left = 0
  right = len(heights) - 1

  while left < right:
      height = min(heights[left], heights[right])
      width = right - left#8-0
      area = height * width
      max_area = max(max_area, area)
      if heights[left] < heights[right]:
          left += 1
      else:
          right -= 1

  return max_area


heights = [1, 8, 6, 2, 5, 4, 8, 3, 7]
print(max_area_two_pointers(heights))
```

(3) TWO SUM 2

```
def twoSum(numbers, target):
  left, right = 0, len(numbers) - 1
  while left < right:
      current_sum = numbers[left] + numbers[right]

      if current_sum == target:
          return [left+1, right+1]  # Return 1-indexed positions

      elif current_sum < target:
          left += 1  # Move the left pointer to the right to increase the sum

      else:
          right -= 1  # Move the right pointer to the left to decrease the sum

# Example usage:
print(twoSum([2, 7, 11, 15], 9))
print(twoSum([1, 3, 4, 5, 7, 10, 11], 9))
```

(4) THREE SUM

```
def three_sum(nums):
  nums.sort()  # Sort the array
  result = []  # Initialize an empty list to store triplets
  for i in range(len(nums) - 2):
    # Skip duplicates for the current number
    if i > 0 and nums[i] == nums[i - 1]:
      continue
    left, right = i + 1, len(nums) - 1
    while left < right:
      total = nums[i] + nums[left] + nums[right]
      if total < 0:
        left += 1
      elif total > 0:
        right -= 1
      else:
        result.append([nums[i], nums[left], nums[right]])
        # Move pointers and skip duplicates
        left += 1
        right -= 1
        while left < right and nums[left] == nums[left + 1]:
          left += 1
        while left < right and nums[right] == nums[right - 1]:
          right -= 1
  return result

# Example usage
nums = [-2, -1, -1, 0, 0, 0, 1, 1, 2]
print(three_sum(nums))
```

(5) MAJORITY ELEMENT

```
def majorityElement(nums):
  # Initialize variables for the candidate and the count
  candidate = None
  count = 0

  # Phase 1: Find the candidate
  for num in nums:
      if count == 0:
          candidate = num

      if num == candidate:
          count += 1
      else:
          count -= 1

  # The candidate is the majority element
  return candidate

# Example usage:
nums1 = [2, 2, 1, 1, 1, 1, 2, 2]
print(majorityElement(nums1))  # Output: 2

nums2 = [3, 2, 4]
print(majorityElement(nums2))  # Output: 3

nums3 = [6, 5, 7]
print(majorityElement(nums3))  # Output: 5
```

(6) MEREGE TWO SORTED ARRAYS IN FIRST ARRAY

```
def mergetwosortedarrays(nums1, m, nums2, n):
    p1 = m - 1
    p2 = n - 1
    p = m + n - 1

    while p1 >= 0 and p2 >= 0:
      if nums1[p1] > nums2[p2]:
          nums1[p] = nums1[p1]
          p1 -= 1
      else:
          nums1[p] = nums2[p2]
          p2 -= 1
      p -= 1
    while p2 >= 0:
        nums1[p] = nums2[p2]
        p2 -= 1
        p -= 1

nums1 = [3, 2, 3, 0, 0, 0]
m = 3
nums2 = [2, 5, 6]
n = 3
mergetwosortedarrays(nums1, m, nums2, n)
print(nums1)
```

(7) REMOVE DUPLICATES FROM SORTED ARRAYS

```
def removeduplicates(numbers):
    if not numbers:
        return 0

    i = 0
    for j in range(1, len(numbers)):
        if numbers[j] != numbers[i]:
            i += 1
            numbers[i] = numbers[j]


    return i + 1

numbers = [1, 1, 2, 3, 3, 4, 4, 5, 5, 5]
k = removeduplicates(numbers)
print(numbers[:k])
print(numbers[k:])
```

(8) REMOVE ELEMENT

```
def remove(nums,val):
    j = 0
    for i in range(len(nums)):
        if nums[i] != val:
            nums[j] = nums[i]
            j += 1

    return j
nums = [3,2,2,3]
val = 2
```

```
new_length = remove(nums, val)
print(new_length)  # New length of the list
print(nums[:new_length])
```