1. def process\_list(lst):

if not lst:

return lst

return sorted(lst)

test\_cases = [

([], []),

([1], [1]),

([7, 7, 7, 7], [7, 7, 7, 7]),

([-5, -1, -3, -2, -4], [-5, -4, -3, -2, -1])

]

for i, (input\_list, expected) in enumerate(test\_cases, 1):

output = process\_list(input\_list)

print(f"Test Case {i}: Input = {input\_list}, Output = {output}, Expected = {expected}")

**Output**: Test Case 1: Input = [], Output = [], Expected = []

Test Case 2: Input = [1], Output = [1], Expected = [1]

Test Case 3: Input = [7, 7, 7, 7], Output = [7, 7, 7, 7], Expected = [7, 7, 7, 7]

Test Case 4: Input = [-5, -1, -3, -2, -4], Output = [-5, -4, -3, -2, -1], Expected = [-5, -4, -3, -2, -1]

2. def selection\_sort(arr):

n = len(arr)

for i in range(n):

min\_index = i

for j in range(i + 1, n):

if arr[j] < arr[min\_index]:

min\_index = j

arr[i], arr[min\_index] = arr[min\_index], arr[i]

return arr

arr = [5, 2, 9, 1, 5, 6]

print(selection\_sort(arr))

**Output:** [1, 2, 5, 5, 6, 9].

3. def bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False

for j in range(n - 1 - i):

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

swapped = True

if not swapped:

break

return arr

arr = [5, 2, 9, 1, 5, 6]

print(bubble\_sort(arr))

**Output:** [1, 2, 5, 5, 6, 9].

4. def bubble\_sort(arr):

n = len(arr)

for i in range(n):

swapped = False # Track if any swaps occur

for j in range(n - 1 - i):

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

swapped = True

if not swapped:

break

return arr

bubble\_test\_cases = [

([64, 25, 12, 22, 11], [11, 12, 22, 25, 64]),

([29, 10, 14, 37, 13], [10, 13, 14, 29, 37]),

([3, 5, 2, 1, 4], [1, 2, 3, 4, 5]),

([1, 2, 3, 4, 5], [1, 2, 3, 4, 5]),

([5, 4, 3, 2, 1], [1, 2, 3, 4, 5])

]

print("Bubble Sort Test Cases:")

for i, (input\_list, expected) in enumerate(bubble\_test\_cases, 1):

output = bubble\_sort(input\_list[:])

print(f"Test {i}: Input = {input\_list}, Output = {output}, Expected = {expected}")

**Output:**

Bubble Sort Test Cases:

Test 1: Input = [64, 25, 12, 22, 11].

**5.** def find\_kth\_missing(arr, k):

left, right = 0, len(arr) - 1

while left <= right:

mid = (left + right) // 2

missing\_count = arr[mid] - (mid + 1)

if missing\_count < k:

left = mid + 1

else:

right = mid - 1

return left + k

arr = [2, 3, 4, 7, 11]

k = 5

print(find\_kth\_missing(arr, k))

**Output:** 9

6. def findPeakElement(nums):

left, right = 0, len(nums) - 1

while left < right:

mid = left + (right - left) // 2

if nums[mid] < nums[mid + 1]:

left = mid + 1

else:

right = mid

return left

**Output: 2.**

**7.** def strStr(haystack: str, needle: str) -> int:

if not needle:

return 0

h\_len, n\_len = len(haystack), len(needle)

if n\_len > h\_len:

return -1

for i in range(h\_len - n\_len + 1):

if haystack[i:i+n\_len] == needle:

return i

return -1

**Output: 0.**

**8.** def substringInWords(words):

result = []

for i in range(len(words)):

for j in range(len(words)):

if i != j and words[i] in words[j]:

result.append(words[i])

break

return list(set(result))

words = ["mass", "as", "hero", "superhero"]

output = substringInWords(words)

print(output)

**Output:** ['as', 'hero']