1.Remove Element:

def remove \_ element(nums, val):

k = 0

for num in nums:

if num != val:

nums[k] = num

k += 1

return k

nums = [3, 2, 2, 3]

val = 3

k = remove\_element(nums, val)

print(f"Output: {k}, nums = {nums[:k]}")

2.LENGTH OF LAST WORD:

def length\_of\_last\_word(s):

words = s.split()

if len(words) == 0:

return 0

return len(words[-1])

# Test the function

s = "Hello World"

print(length\_of\_last\_word(s))

3. max subarray:

def max\_subarray\_sum(nums):

max\_sum = float('-inf')

for i in range(len(nums)):

current\_sum = 0

for j in range(i, len(nums)):

current\_sum += nums[j]

max\_sum = max(max\_sum, current\_sum)

return max\_sum

# Test the function

nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

print(max\_subarray\_sum(nums))

4. COMBINATION SUM:

def combinationSum(candidates, target):

def backtrack(start, path, target):

if target == 0:

result.append(path[:])

return

for i in range(start, len(candidates)):

if candidates[i] > target:

continue

path.append(candidates[i])

backtrack(i, path, target - candidates[i])

path.pop()

candidates.sort()

result = []

backtrack(0, [], target)

return result

# Test the function with the provided example

candidates = [2, 3, 6, 7]

target = 7

print(combinationSum(candidates, target))

5. permutation II:

from itertools import permutations

def unique\_permutations(nums):

return list(set(permutations(nums)))

# Test the function with the given example

nums = [1, 1, 2]

print(unique\_permutations(nums))