

ASSIGNMENT 2:

11. Container With Most Water You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the

maximum amount of water a container can store. Notice that you may not slant the container.

CODE:

```
def maxArea(A, Len):
    area = 0
    for i in range(Len):
        for j in range(i + 1, Len):
            area = max(area, min(A[j], A[i]) * (j - i))
    return area
a = [ 1, 5, 4, 3 ]
b = [ 3, 1, 2, 4, 5 ]
len1 = len(a)
print(maxArea(a, len1))
len2 = len(b)
print(maxArea(b, len2))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
1
1
Process finished with exit code 0
```

12. Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value
I 1 V 5 X 10 L 50 C 100 D 500 M 1000 For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is

XX + V + II. Roman numerals are usually written largest to smallest from left to right. However, the numeral

for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are

six instances where subtraction is used: ● I can be placed before V (5) and X (10) to make 4 and 9. ● X can be placed before L (50) and C (100) to make 40 and 90. ● C can be placed before D (500) and M (1000) to make 400 and 900. Given an integer, convert it to a roman numeral.(1001)

Program:

```
def int_to_roman(num):
    val = [
        1000, 900, 500, 400,
        100, 90, 50, 40,
        10, 9, 5, 4,
        1
    ]
    syms = [
        "M", "CM", "D", "CD",
        "C", "XC", "L", "XL",
        "X", "IX", "V", "IV",
        "I"
    ]
    roman_numeral = ""
    i = 0
    while num > 0:
        for _ in range(num // val[i]):
```

```

        roman_numeral += syms[i]
        num -= val[i]
        i += 1
    return roman_numeral
print(int_to_roman(1001))

```

Output:

```

C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
MI

Process finished with exit code 0

```

13. Roman to Integer Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value I 1 V 5 X 10 L 50 C 100 D 500 M 1000 For example, 2 is written as II in Roman numeral,

just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II. Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used: • I can be placed before V (5) and X (10)

to make 4 and 9. • X can be placed before L (50) and C (100) to make 40 and 90. • C can be placed before D (500) and M (1000) to make 400 and 900.

Program:

```

def roman_to_int(roman):
    roman_values = {
        'I': 1,
        'V': 5,
        'X': 10,
        'L': 50,
        'C': 100,
        'D': 500,
        'M': 1000
    }
    total = 0
    prev_value = 0
    for symbol in reversed(roman):
        value = roman_values[symbol]
        if value < prev_value:
            total -= value
        else:
            total += value
        prev_value = value
    return total
print(roman_to_int('XXVII'))

```

Output:

```

C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
27

Process finished with exit code 0

```

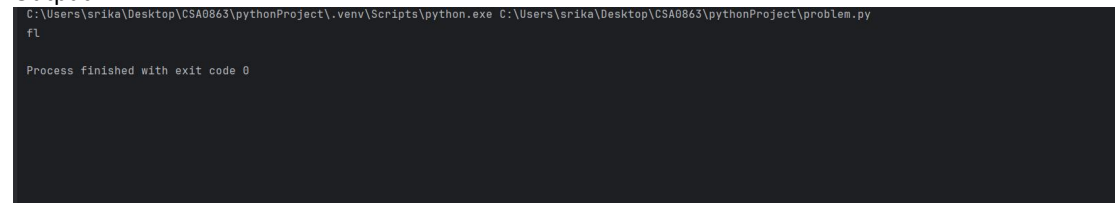
14. Longest Common Prefix Write a function to find the longest common prefix string amongst an array

of strings. If there is no common prefix, return an empty string ""

Program:

```
def longest_common_prefix(strs):
    if not strs:
        return ""
    for i, char in enumerate(strs[0]):
        if any(i == len(s) or s[i] != char for s in strs[1:]):
            return strs[0][:i]
    return strs[0]
str_list = ["flower", "flow", "flight"]
print(longest_common_prefix(str_list))
```

Output:



15. 3Sum Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that $i \neq j$, i

$\neq k$, and $j \neq k$, and $\text{nums}[i] + \text{nums}[j] + \text{nums}[k] == 0$. Notice that the solution set must not contain duplicate triplets. Example 1: Input: $\text{nums} = [-1, 0, 1, 2, -1, -4]$ Output: $[[-1, -1, 2], [-1, 0, 1]]$ Explanation: $\text{nums}[0] + \text{nums}[1] + \text{nums}[2] = (-1) + 0 + 1 = 0$. $\text{nums}[1] + \text{nums}[2] + \text{nums}[4] = 0 + 1 + (-1) = 0$. $\text{nums}[0]$

$+ \text{nums}[3] + \text{nums}[4] = (-1) + 2 + (-1) = 0$. The distinct triplets are $[-1, 0, 1]$ and $[-1, -1, 2]$. Notice that the order of the output and the order of the triplets does not matter.

Program:

```
def three_sum(nums):
    nums.sort()
    result = []

    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        left = i + 1
        right = len(nums) - 1
        while left < right:
            total = nums[i] + nums[left] + nums[right]
            if total == 0:
                result.append([nums[i], nums[left], nums[right]])
                # Skip duplicate elements
                while left < right and nums[left] == nums[left + 1]:
                    left += 1
                while left < right and nums[right] == nums[right - 1]:
                    right -= 1
                left += 1
                right -= 1
            elif total < 0:
                left += 1
            else:
                right -= 1
    return result
nums = [-1, 0, 1, 2, -1, -4]
print(three_sum(nums))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
[[-1, -1, 2], [-1, 0, 1]]

Process finished with exit code 0
```

16. 3Sum Closest Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to `target`. Return the sum of the three integers. You may assume that each input would have exactly one solution.

Program:

```
def three_sum_closest(nums, target):
    nums.sort() # Sorting the array
    closest_sum = float('inf')
    for i in range(len(nums) - 2):
        left = i + 1
        right = len(nums) - 1
        while left < right:
            total = nums[i] + nums[left] + nums[right]
            if abs(total - target) < abs(closest_sum - target):
                closest_sum = total
            if total < target:
                left += 1
            elif total > target:
                right -= 1
            else:
                return target
    return closest_sum
nums = [-1, 2, 1, -4]
target = 1
print(three_sum_closest(nums, target))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
2

Process finished with exit code 0
```

17. Letter Combinations of a Phone Number Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order. A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Program:

```
def letter_combinations(digits):
    if not digits:
        return []
    digit_to_letters = {
        '2': 'abc',
        '3': 'def',
        '4': 'ghi',
        '5': 'jkl',
        '6': 'mno',
        '7': 'pqrs',
```

```

        '8': 'tuv',
        '9': 'wxyz'
    }
    def backtrack(combination, next_digits):
        if len(next_digits) == 0:
            result.append(combination)
        else:
            for letter in digit_to_letters[next_digits[0]]:
                backtrack(combination + letter, next_digits[1:])

    result = []
    backtrack("", digits)
    return result
digits = "23"
print(letter_combinations(digits))

```

Output:

```

C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
['ad', 'ae', 'af', 'bd', 'be', 'bf', 'cd', 'ce', 'cf']

Process finished with exit code 0

```

18.) 4Sum Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that: • $0 \leq a, b, c, d < n$ • a, b, c, and d are distinct. • $nums[a] + nums[b] + nums[c] + nums[d] == target$

Program:

```

def four_sum(nums, target):
    nums.sort()
    n = len(nums)
    quadruplets = []
    for i in range(n - 3):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        for j in range(i + 1, n - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue
            left = j + 1
            right = n - 1
            while left < right:
                total = nums[i] + nums[j] + nums[left] + nums[right]
                if total == target:
                    quadruplets.append([nums[i], nums[j], nums[left], nums[right]])
                    while left < right and nums[left] == nums[left + 1]:
                        left += 1
                    while left < right and nums[right] == nums[right - 1]:
                        right -= 1
                    left += 1
                    right -= 1
                elif total < target:
                    left += 1
                else:
                    right -= 1
            return quadruplets
nums = [1, 0, -1, 0, -2, 2]
target = 0
print(four_sum(nums, target))

```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]

Process finished with exit code 0
```

19. Remove Nth Node From End of List Given the head of a linked list, remove the nth node from the end of the list and return its head.

Program:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def remove_nth_from_end(head, n):
    dummy = ListNode(0)
    dummy.next = head
    first = dummy
    second = dummy
    for _ in range(n + 1):
        first = first.next
    while first:
        first = first.next
        second = second.next
    second.next = second.next.next
    return dummy.next
head = ListNode(1)
head.next = ListNode(2)
head.next.next = ListNode(3)
head.next.next.next = ListNode(4)
head.next.next.next.next = ListNode(5)
n = 2
new_head = remove_nth_from_end(head, n)
while new_head:
    print(new_head.val, end=" -> ")
    new_head = new_head.next
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
1 -> 2 -> 3 -> 5 ->
Process finished with exit code 0
```

20. Valid Parentheses Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if

the input string is valid. An input string is valid if: 1. Open brackets must be closed by the same type of brackets. 2. Open brackets must be closed in the correct order. 3. Every close bracket has a corresponding open bracket of the same type.

Program:

```
def is_valid(s):
    stack = []
    mapping = {'(': ')', '{': '}', '[': ']'}
    for char in s:
        if char in mapping.values():
            stack.append(char)
```

```
        elif char in mapping.keys():
            if not stack or mapping[char] != stack.pop():
                return False
        return not stack
print(is_valid("("))
print(is_valid "()[]{}"))
print(is_valid("()"))
print(is_valid("([])"))
print(is_valid("{}"))
```

Output:

```
C:\Users\srika\Desktop\CSA0863\pythonProject\.venv\Scripts\python.exe C:\Users\srika\Desktop\CSA0863\pythonProject\problem.py
True
True
False
False
True

Process finished with exit code 0
```