Assignment-2

Name: G.venkatapraveen

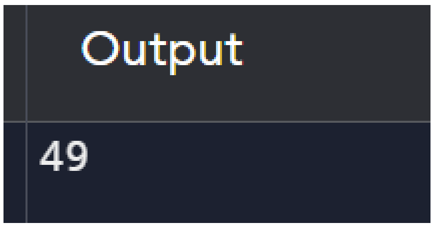
Reg.no: 192373023

Dept: Cse(D.s)

11. Container With Most Water

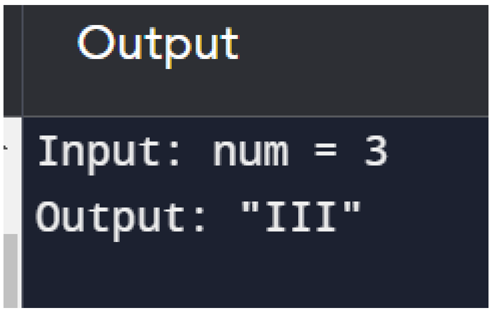
Program:

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

   while left < right:
      width = right - left
      h = min(height[left], height[right])
      max_area = max(max_area, width * h)

      if height[left] < height[right]:
         left += 1
      else:
         right -= 1

   return max_area

# Example
height = [1, 8, 6, 2, 5, 4, 8, 3, 7]
print(max_area(height))  # Output: 49
```


```
Output
49
```

12. Integer to Roman

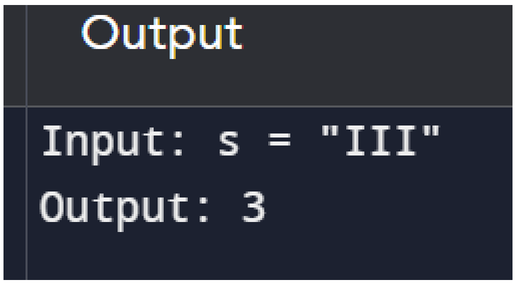
```
def int_to_roman(num):
    if not isinstance(num, int) or num <= 0 or num > 3999:
        raise ValueError("Input must be a positive integer between 1 and 3999")

    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_num = ''
    i = 0

    while num > 0:
        for _ in range(num // val[i]):
            roman_num += syms[i]
            num -= val[i]
        i += 1

    return roman_num

# Test the function
try:
    num = 3
    roman_numeral = int_to_roman(num)
    print(f"Input: {num}\nOutput: {roman_numeral}")
except ValueError as e:
    print(f"Error: {e}")
```



```
Input: num = 3
Output: "III"
```

13. Roman to Integer

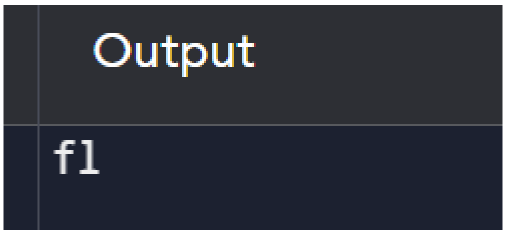
```python
def roman_to_int(s: str) -> int:
    roman_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    result = 0
    prev_value = 0

    for char in s:
        value = roman_dict[char]
        if value > prev_value:
            result += value - 2 * prev_value
        else:
            result += value
        prev_value = value

    return result

# Test the function
input_roman = "III"
output_integer = roman_to_int(input_roman)
print(f"Input: {input_roman}")
print(f"Output: {output_integer}")
```



```
Input: s = "III"
Output: 3
```

14. Longest Common Prefix

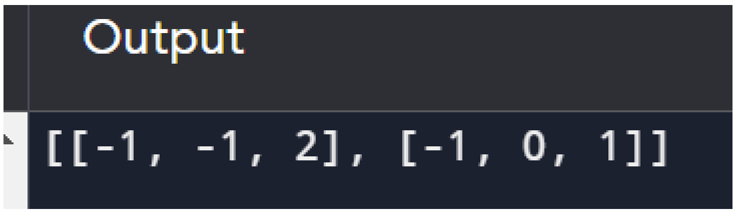
```
def longest_common_prefix(strs):
    if not strs:
        return ""

    # Sort the list of strings
    strs.sort()

    # Find the common prefix between the first and last strings
    prefix = ""
    for i in range(len(strs[0])):
        if strs[0][i] == strs[-1][i]:
            prefix += strs[0][i]
        else:
            break

    return prefix

# Test the function
strs = ["flower", "flow", "flight"]
output = longest_common_prefix(strs)
print(output)  # Output: "fl".
```



```
Output
f1
```

15. 3Sum

```
def find_triplets(nums):
    nums.sort()
    triplets = []
    n = len(nums)

    for i in range(n-2):
        if i > 0 and nums[i] == nums[i-1]:
            continue

        left, right = i+1, n-1

        while left < right:
            total = nums[i] + nums[left] + nums[right]

            if total < 0:
                left += 1
            elif total > 0:
                right -= 1
            else:
                triplets.append([nums[i], 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

    return triplets

# Example
nums = [-1, 0, 1, 2, -1, -4]
result = find_triplets(nums)
print(result)  # Output: [[-1, -1, 2], [-1, 0, 1]]
```

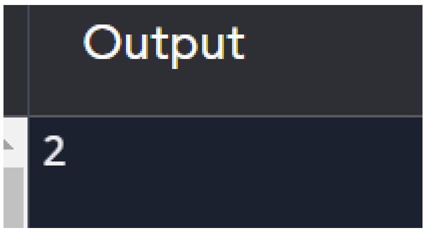


```
[[-1, -1, 2], [-1, 0, 1]]
```

16. 3Sum Closest

```
def threeSumClosest(nums, target):
    nums.sort()
    n = len(nums)
    closest_sum = float('inf')

    for i in range(n):
        left, right = i + 1, n - 1
        while left < right:
            current_sum = nums[i] + nums[left] + nums[right]
            if abs(target - current_sum) < abs(target - closest_sum):
                closest_sum = current_sum
            if current_sum < target:
                left += 1
            else:
                right -= 1

    return closest_sum
```

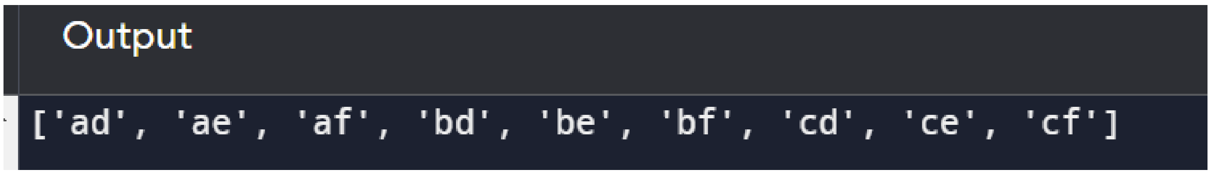
```
# Example
nums = [-1, 2, 1, -4]
target = 1
result = threeSumClosest(nums, target)
print(result)  # Output: 2
```



```
Output
2
```

17. Letter Combinations of a Phone Number

```
from typing import List


def letter_combinations(digits: str) -> List[str]:
    if not digits:
        return []

    phone_mapping = {
        '2': 'abc',
        '3': 'def',
        '4': 'ghi',
        '5': 'jkl',
        '6': 'mno',
        '7': 'pqrs',
        '8': 'tuv',
        '9': 'wxyz'
    }

    def backtrack(index, path):
        if index == len(digits):
            combinations.append(''.join(path))
            return

        for char in phone_mapping[digits[index]]:
            path.append(char)
            backtrack(index + 1, path)
            path.pop()

    combinations = []
    backtrack(0, [])

    return combinations

# Test the function with the provided example
digits = "23"
output = letter_combinations(digits)
print(output)
```

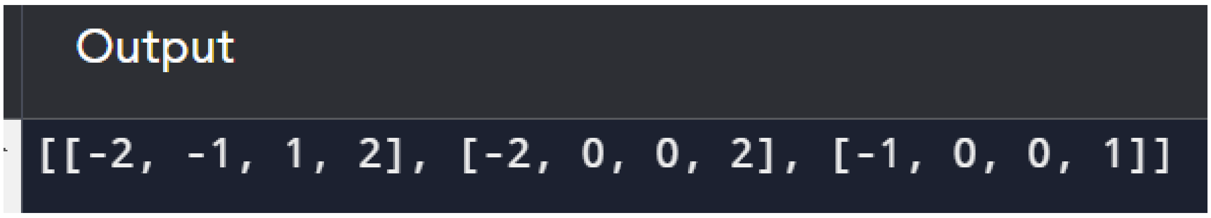

Output

```
['ad', 'ae', 'af', 'bd', 'be', 'bf', 'cd', 'ce', 'cf']
```

18. 4Sum

```
def fourSum(nums, target):
    nums.sort()
    result = []
    n = len(nums)

    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, right = j + 1, n - 1
        while left < right:
            total = nums[i] + nums[j] + nums[left] + nums[right]
            if total == target:
                result.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 result

# Test the function with the provided example
nums = [1, 0, -1, 0, -2, 2]
target = 0
output = fourSum(nums, target)
print(output)  # Output: [[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]
```


Output

```
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]
```

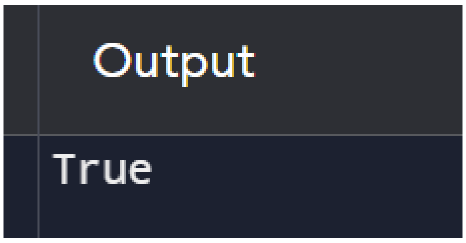
19. Remove Nth Node From End of List

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def removeNthFromEnd(head, n):
    dummy = ListNode(0)
    dummy.next = head
    first = dummy
    second = dummy

    for _ in range(n + 1):
        first = first.next

    while first is not None:
        first = first.next
        second = second.next

    second.next = second.next.next

    return dummy.next

# Example
head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))
n = 2
result = removeNthFromEnd(head, n)
```

```
# Print the modified linked list
while result:
    print(result.val, end=" ")
    result = result.next
```

20. Valid Parentheses

```
def is_valid(s: str) -> bool:
    stack = []
    mapping = {')': '(', '}': '{', ']': '['}

    for char in s:
        if char in mapping:
            top_element = stack.pop() if stack else '#'
            if mapping[char] != top_element:
                return False
        else:
            stack.append(char)

    return not stack

# Test the function with the provided example
input_string = "()"
print(is_valid(input_string))  # Output: True
```



```
True
```