main.py

Run

Output

```python
def maxArea(height):
    max_water = 0
    left = 0
    right = len(height) - 1

    while left < right:
        water = (right - left) * min(height[left], height[right])
        max_water = max(max_water, water)

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

    return max_water

# Example usage:
height = [1,8,6,2,5,4,8,3,7]
print("Maximum amount of water is:", maxArea(height))
```

```
Maximum amount of water is: 49

=== Code Execution Successful ===
```

ain.py

```python
def intToRoman(num):
    val = [
        1000, 900, 500, 400,
        100, 90, 50, 40,
        10, 9, 5, 4,
        1
    ]
    syb = [
        "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 += syb[i]
            num -= val[i]
        i += 1
    return roman_num

# Example usage:
num = 3
print("Roman numeral for", num, "is", intToRoman(num))
```

Output

```
Roman numeral for 3 is III

=== Code Execution Successful ===
```

main.py

Run

```python
def romanToInt(s):
    roman_dict = {
        "I": 1,
        "V": 5,
        "X": 10,
        "L": 50,
        "C": 100,
        "D": 500,
        "M": 1000
    }
    total = 0
    for i in range(len(s) - 1):
        if roman_dict[s[i]] < roman_dict[s[i + 1]]:
            total -= roman_dict[s[i]]
        else:
            total += roman_dict[s[i]]
    total += roman_dict[s[-1]]
    return total

# Example usage:
s = "IV"
print("Integer for", s, "is", romanToInt(s))
```

Output

```
Integer for IV is 4

=== Code Execution Successful ===
```

in.py

Run

Output

```python
def longestCommonPrefix(strs):
    if not strs:
        return ""
    prefix = strs[0]
    for s in strs[1:]:
        while not s.startswith(prefix):
            prefix = prefix[:-1]
            if not prefix:
                return ""
    return prefix

# Example usage:
strs = ["flower","flow","flight"]
print("Longest common prefix is:", longestCommonPrefix(strs))
```

```
Longest common prefix is: fl

=== Code Execution Successful ===
```

ain.py

Save Run

```python
def threeSum(nums):
    nums.sort()
    n = len(nums)
    result = []
    for i in range(n-2):
        if i > 0 and nums[i] == nums[i-1]:
            continue
        left = i + 1
        right = n - 1
        while left < right:
            total = nums[i] + nums[left] + nums[right]
            if total == 0:
                result.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
            elif total < 0:
                left += 1
            else:
                right -= 1
    return result

# Example usage:
nums = [-1, 0, 1, 2, -1, -4]
print(threeSum(nums))
```

Output

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

=== Code Execution Successful ===
```

```python
def threeSumClosest(nums, target):
    nums.sort()
    closest = float('inf')
    for i in range(len(nums) - 2):
        left, right = i + 1, len(nums) - 1
        while left < right:
            total = nums[i] + nums[left] + nums[right]
            if total == target:
                return total
            if abs(total - target) < abs(closest - target):
                closest = total
            if total < target:
                left += 1
            else:
                right -= 1
    return closest

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

```
2

=== Code Execution Successful ===
```

ain.py

Save Run

```python
def letterCombinations(digits):
    if not digits:
        return []

    phone = {'2': ['a', 'b', 'c'],
             '3': ['d', 'e', 'f'],
             '4': ['g', 'h', 'i'],
             '5': ['j', 'k', 'l'],
             '6': ['m', 'n', 'o'],
             '7': ['p', 'q', 'r', 's'],
             '8': ['t', 'u', 'v'],
             '9': ['w', 'x', 'y', 'z']}

    def backtrack(combination, next_digits):
        if len(next_digits) == 0:
            output.append(combination)
        else:
            for letter in phone[next_digits[0]]:
                backtrack(combination + letter, next_digits[1:])

    output = []
    backtrack("", digits)
    return output

# Example usage:
digits = "23"
print(letterCombinations(digits))
```

Output

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

=== Code Execution Successful ===
```

in.py

Save Run Output

```python
def fourSum(nums, target):
    nums.sort()
    result = []
    for i in range(len(nums) - 3):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        for j in range(i + 1, len(nums) - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue
            left, right = j + 1, len(nums) - 1
            while left < right:
                curr_sum = nums[i] + nums[j] + nums[left] + nums[right]
                if curr_sum == 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 curr_sum < target:
                    left += 1
                else:
                    right -= 1
    return result

# Example usage:
nums = [2,2,2,2,2]
```

```
Quadruplets that sum up to 8 :
[2, 2, 2, 2]

=== Code Execution Successful ===
```

ain.py

Run

Output

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

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

    # Move the first pointer n steps ahead
    for _ in range(n):
        first = first.next

    # Move both pointers until the first pointer reaches the end
    while first.next:
        first = first.next
        second = second.next

    # Remove the nth node from the end
    second.next = second.next.next

    return dummy.next
```

```
=== Code Execution Successful ===
```

in.py

Run

Output

```python
def isValid(s):
    stack = []
    mapping = {")": "(", "}": "{", "]": "["}

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

    return len(stack) == 0
```

```
=== Code Execution Successful ===
```