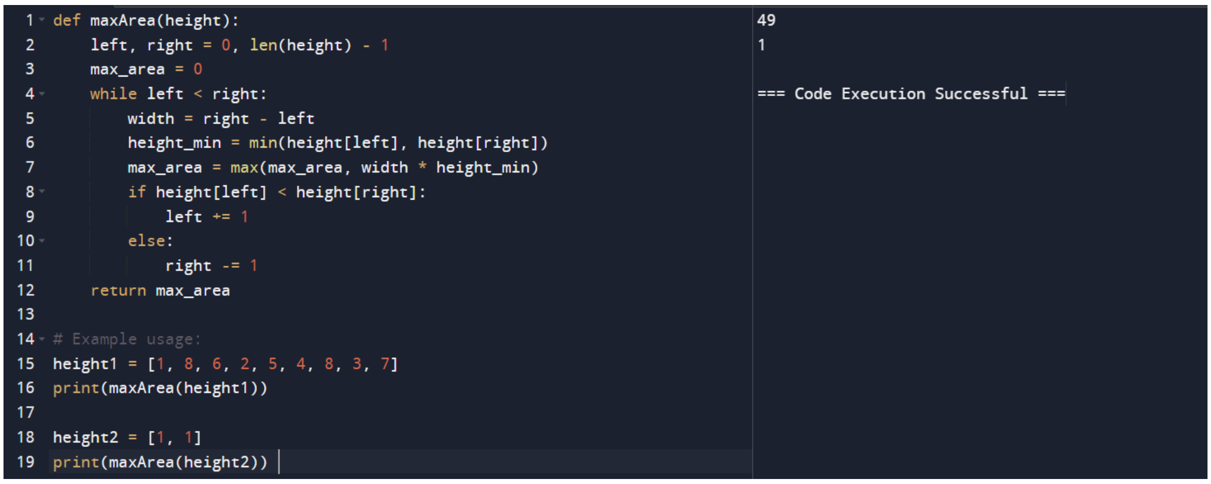
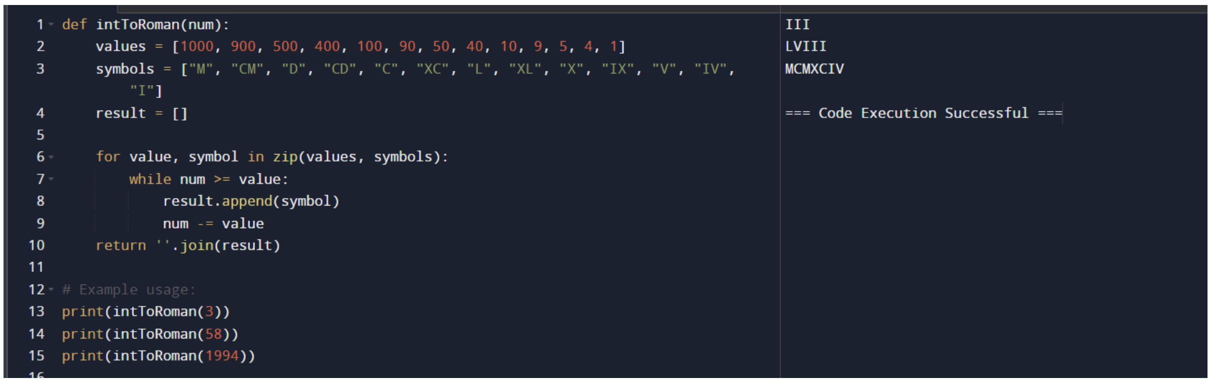
1.



```python
def maxArea(height):
    left, right = 0, len(height) - 1
    max_area = 0
    while left < right:
        width = right - left
        height_min = min(height[left], height[right])
        max_area = max(max_area, width * height_min)
        if height[left] < height[right]:
            left += 1
        else:
            right -= 1
    return max_area

# Example usage:
height1 = [1, 8, 6, 2, 5, 4, 8, 3, 7]
print(maxArea(height1))

height2 = [1, 1]
print(maxArea(height2))
```

```
49
1

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

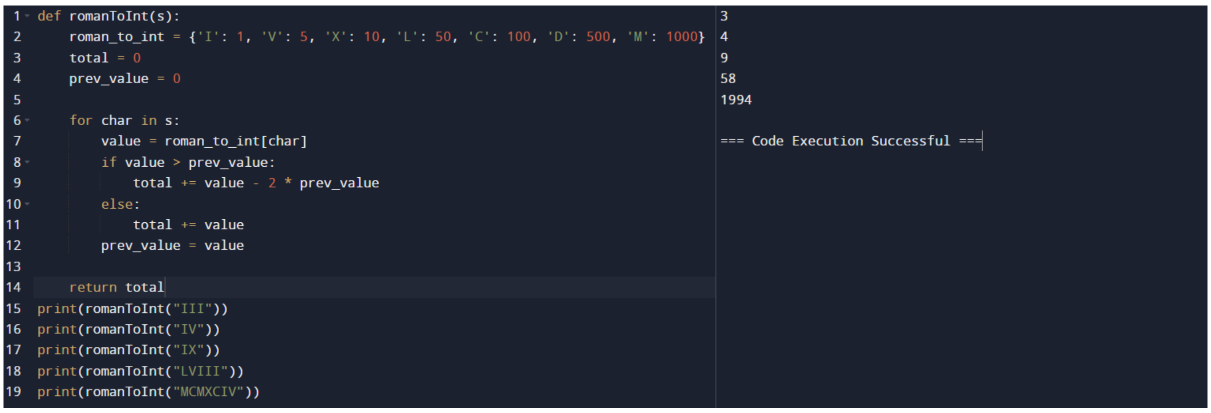
2.



```python
def intToRoman(num):
    values = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]
    symbols = ["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV",
        "I"]
    result = []

    for value, symbol in zip(values, symbols):
        while num >= value:
            result.append(symbol)
            num -= value
    return ''.join(result)

# Example usage:
print(intToRoman(3))
print(intToRoman(58))
print(intToRoman(1994))
```

```
III
LVIII
MCMXCIV

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

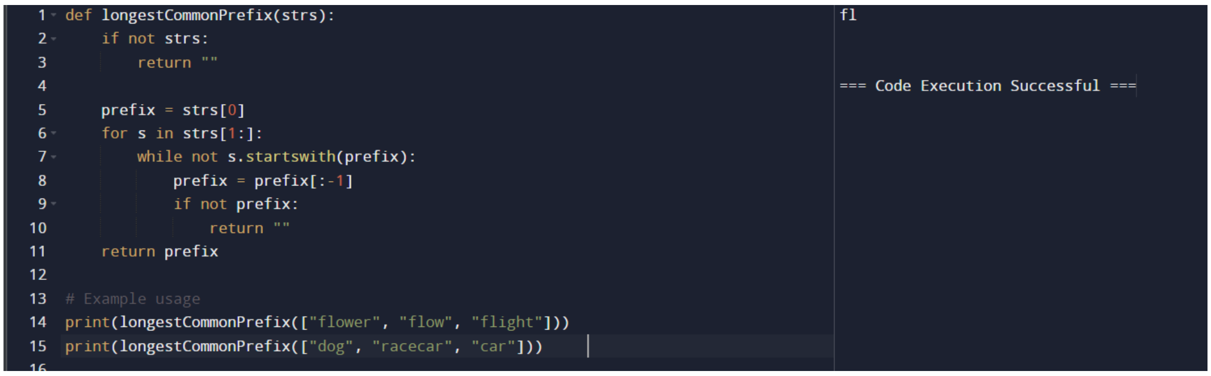
3.



```python
def romanToInt(s):
    roman_to_int = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    total = 0
    prev_value = 0

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

    return total
print(romanToInt("III"))
print(romanToInt("IV"))
print(romanToInt("IX"))
print(romanToInt("LVIII"))
print(romanToInt("MCMXCIV"))
```

```
3
4
9
58
1994

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

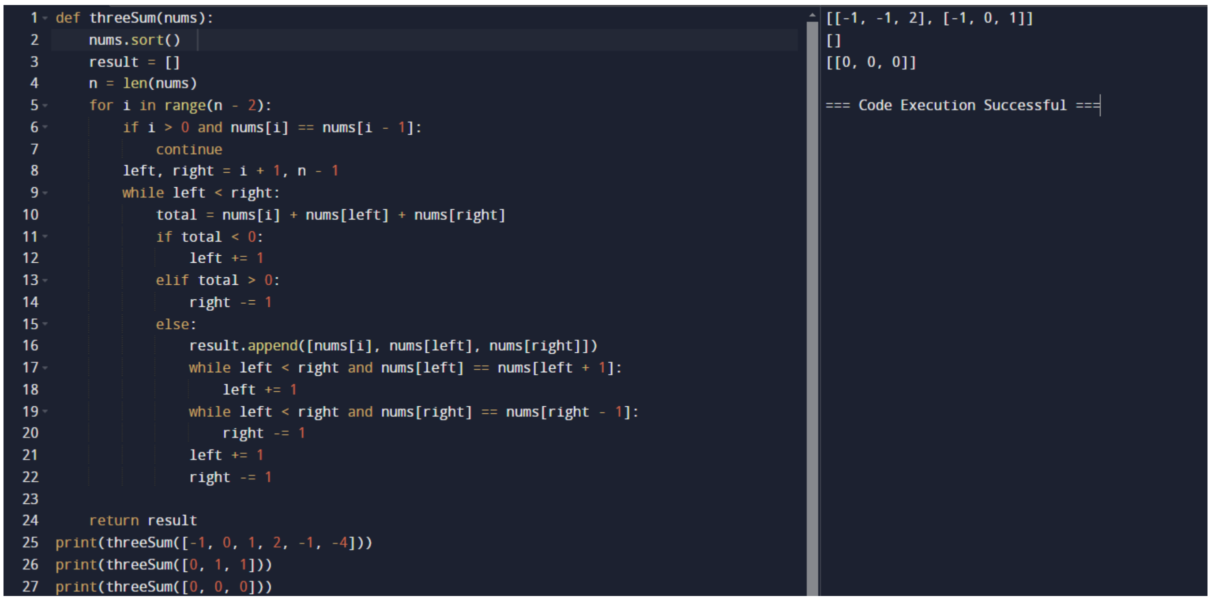
4.



```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
print(longestCommonPrefix(["flower", "flow", "flight"]))
print(longestCommonPrefix(["dog", "racecar", "car"]))
```

```
fl


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

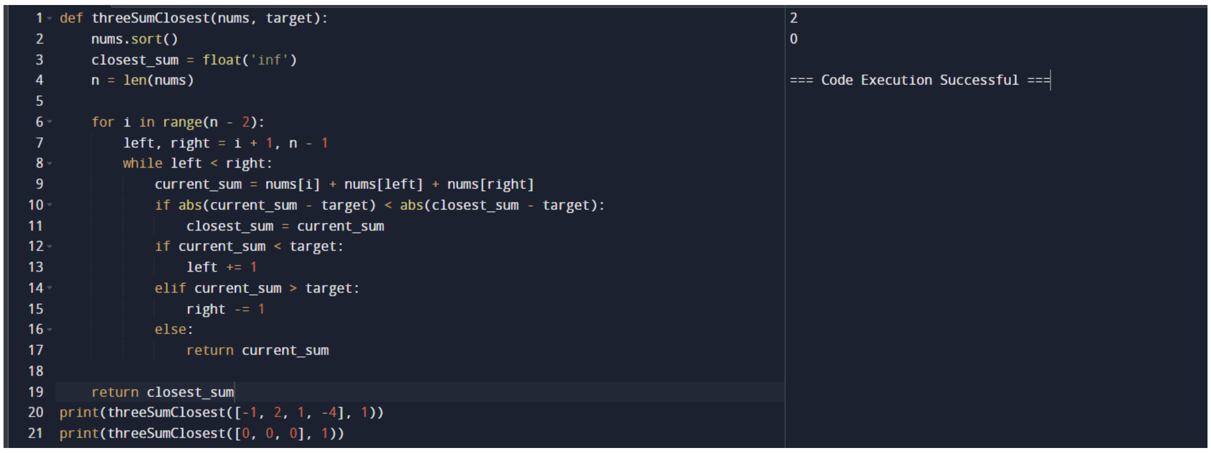
5.



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

    return result
print(threeSum([-1, 0, 1, 2, -1, -4]))
print(threeSum([0, 1, 1]))
print(threeSum([0, 0, 0]))
```

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

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

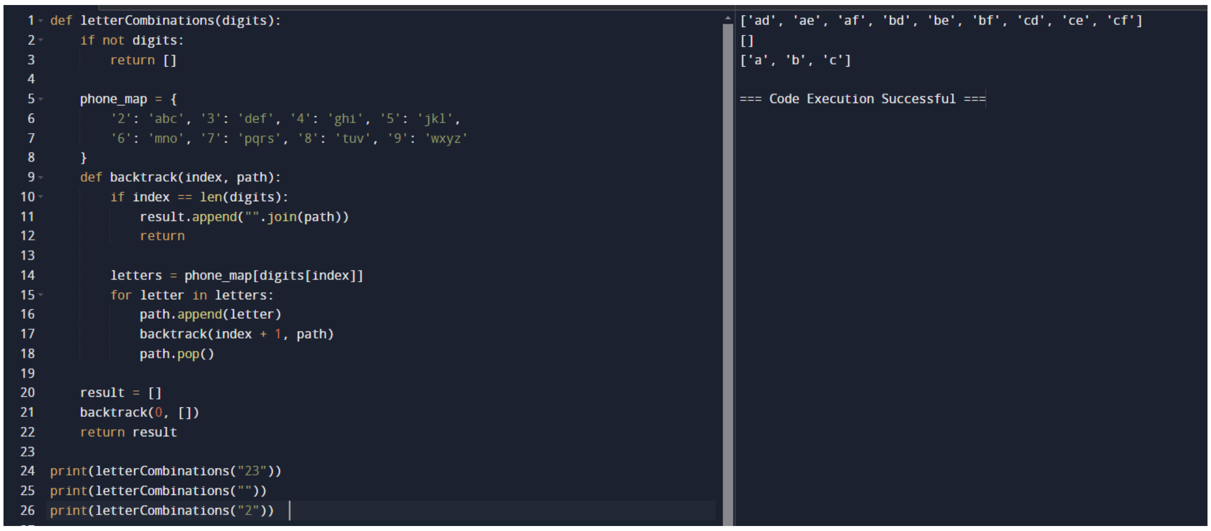
6.



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

    for i in range(n - 2):
        left, right = i + 1, n - 1
        while left < right:
            current_sum = nums[i] + nums[left] + nums[right]
            if abs(current_sum - target) < abs(closest_sum - target):
                closest_sum = current_sum
            if current_sum < target:
                left += 1
            elif current_sum > target:
                right -= 1
            else:
                return current_sum

    return closest_sum
print(threeSumClosest([-1, 2, 1, -4], 1))
print(threeSumClosest([0, 0, 0], 1))
```

```
2
0

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

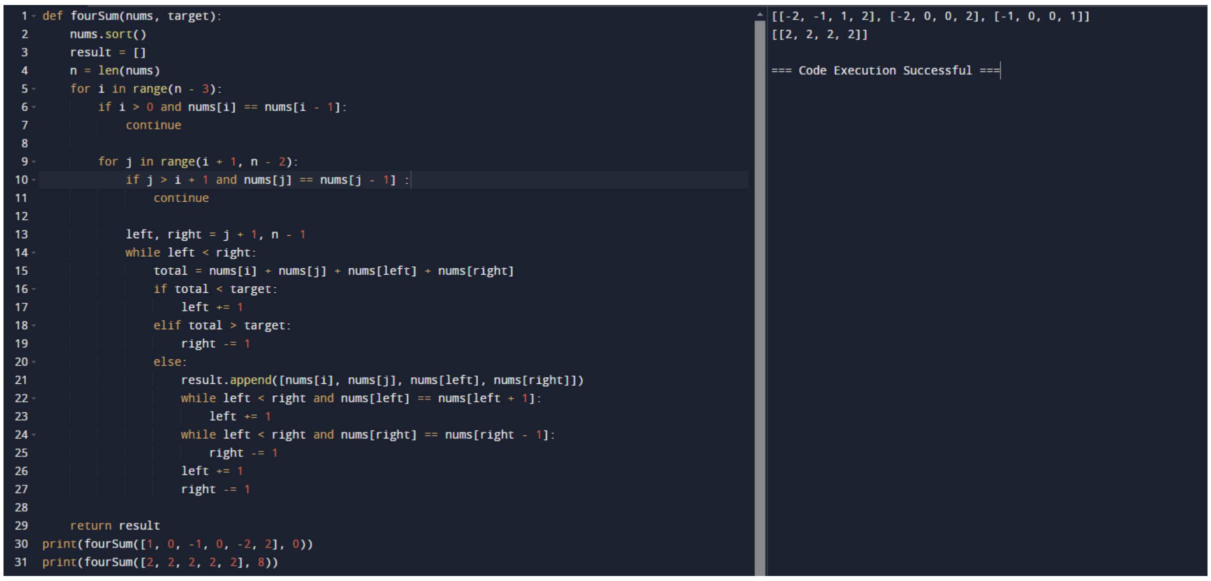
7.



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

    phone_map = {
        '2': 'abc', '3': 'def', '4': 'ghi', '5': 'jkl',
        '6': 'mno', '7': 'pqrs', '8': 'tuv', '9': 'wxyz'
    }
    def backtrack(index, path):
        if index == len(digits):
            result.append("".join(path))
            return

        letters = phone_map[digits[index]]
        for letter in letters:
            path.append(letter)
            backtrack(index + 1, path)
            path.pop()

    result = []
    backtrack(0, [])
    return result

print(letterCombinations("23"))
print(letterCombinations(""))
print(letterCombinations("2"))
```

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

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

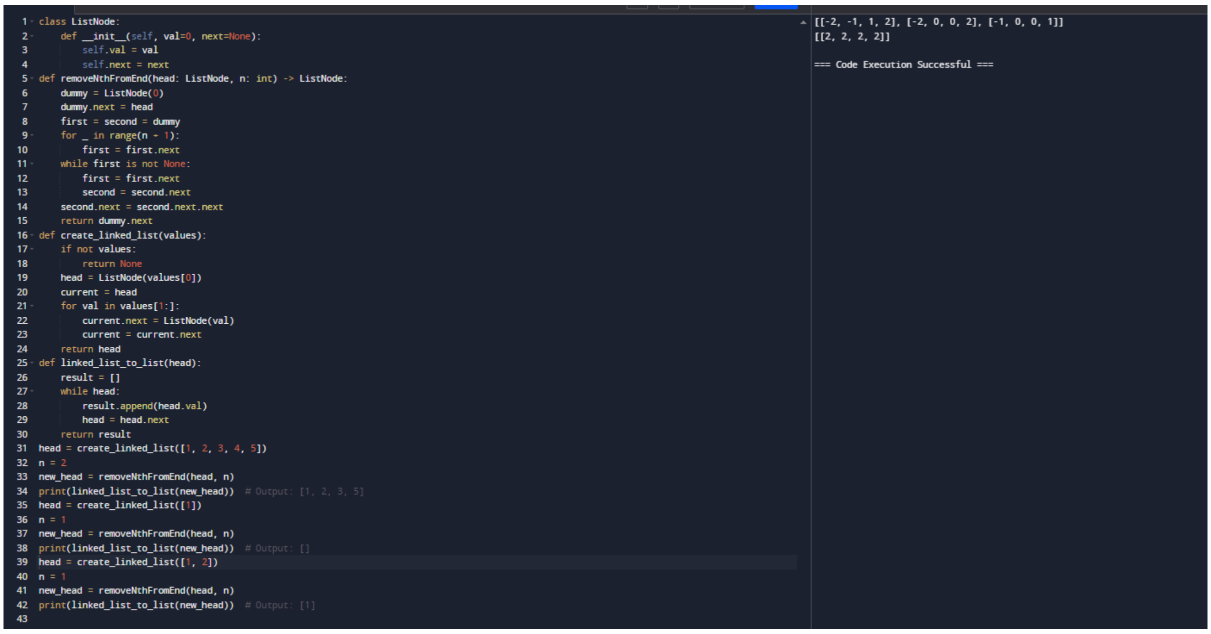
8.



```python
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:
                    left += 1
                elif total > target:
                    right -= 1
                else:
                    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

    return result
print(fourSum([1, 0, -1, 0, -2, 2], 0))
print(fourSum([2, 2, 2, 2, 2], 8))
```

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

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

9.



```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def removeNthFromEnd(head: ListNode, n: int) -> ListNode:
    dummy = ListNode(0)
    dummy.next = head
    first = 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
def create_linked_list(values):
    if not values:
        return None
    head = ListNode(values[0])
    current = head
    for val in values[1:]:
        current.next = ListNode(val)
        current = current.next
    return head
def linked_list_to_list(head):
    result = []
    while head:
        result.append(head.val)
        head = head.next
    return result
head = create_linked_list([1, 2, 3, 4, 5])
n = 2
new_head = removeNthFromEnd(head, n)
print(linked_list_to_list(new_head))  # Output: [1, 2, 3, 5]
head = create_linked_list([1])
n = 1
new_head = removeNthFromEnd(head, n)
print(linked_list_to_list(new_head))  # Output: []
head = create_linked_list([1, 2])
n = 1
new_head = removeNthFromEnd(head, n)
print(linked_list_to_list(new_head))  # Output: [1]

```

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

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