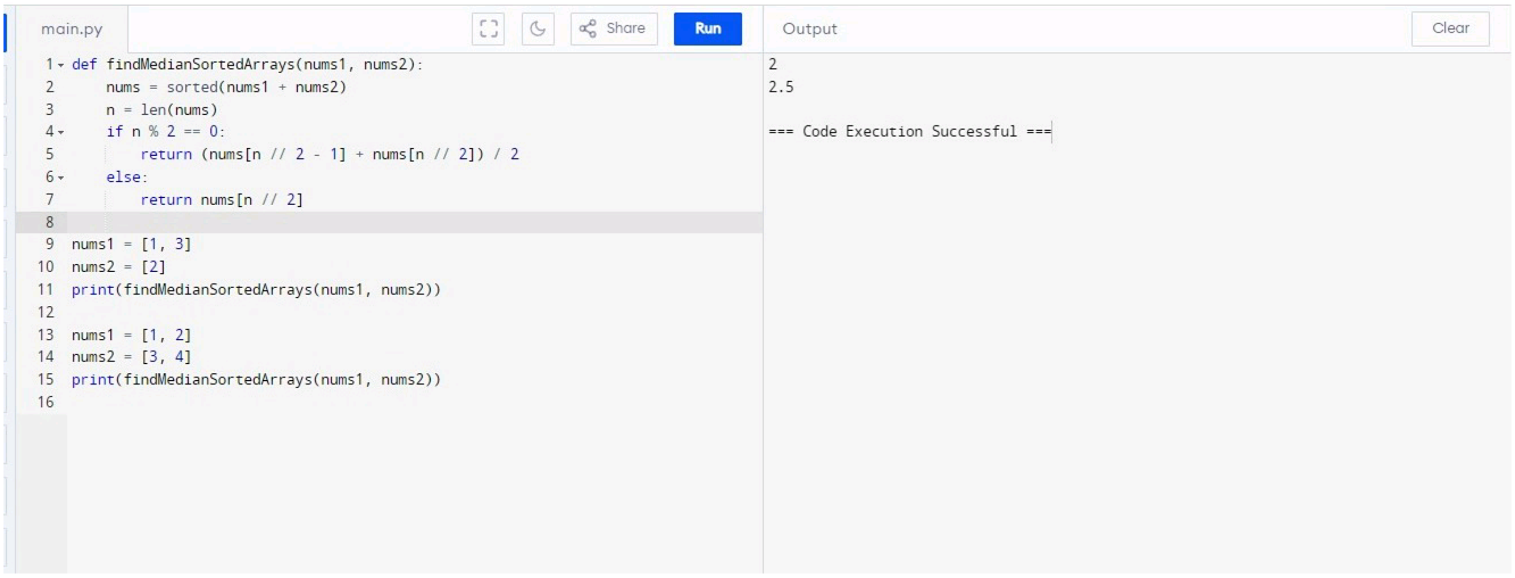
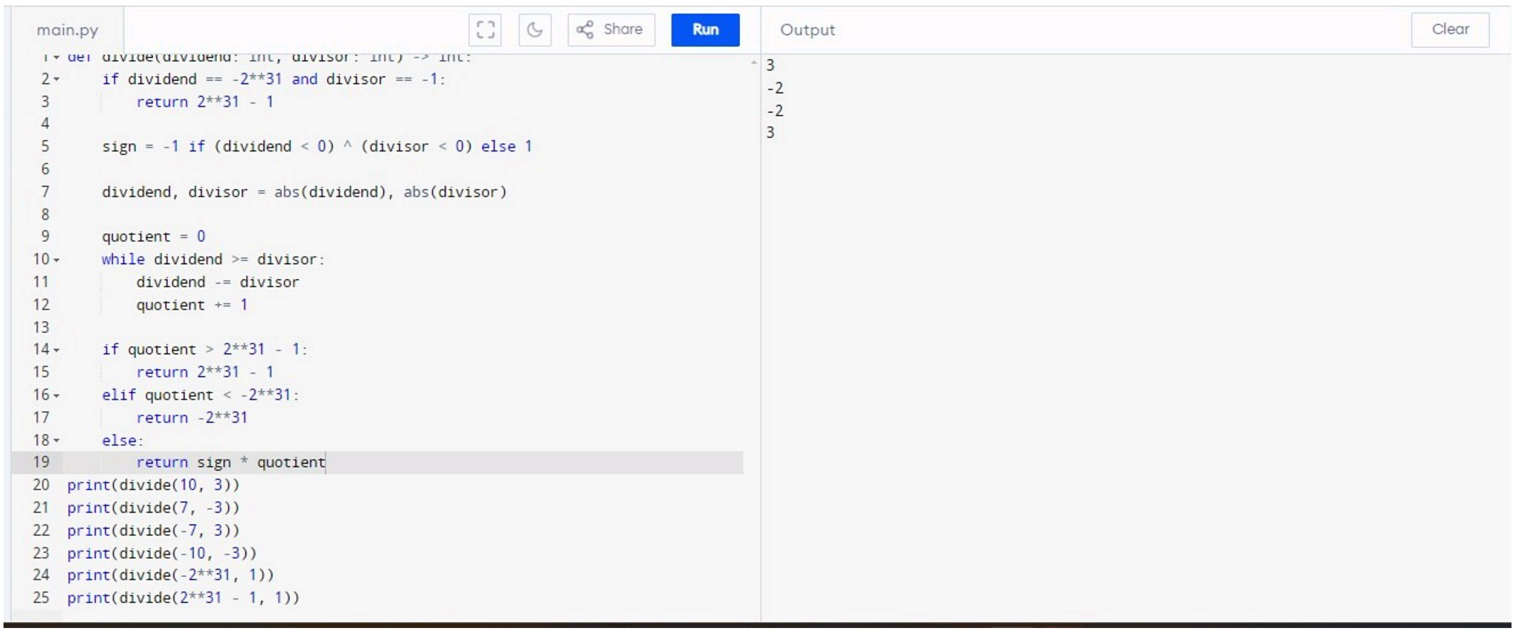


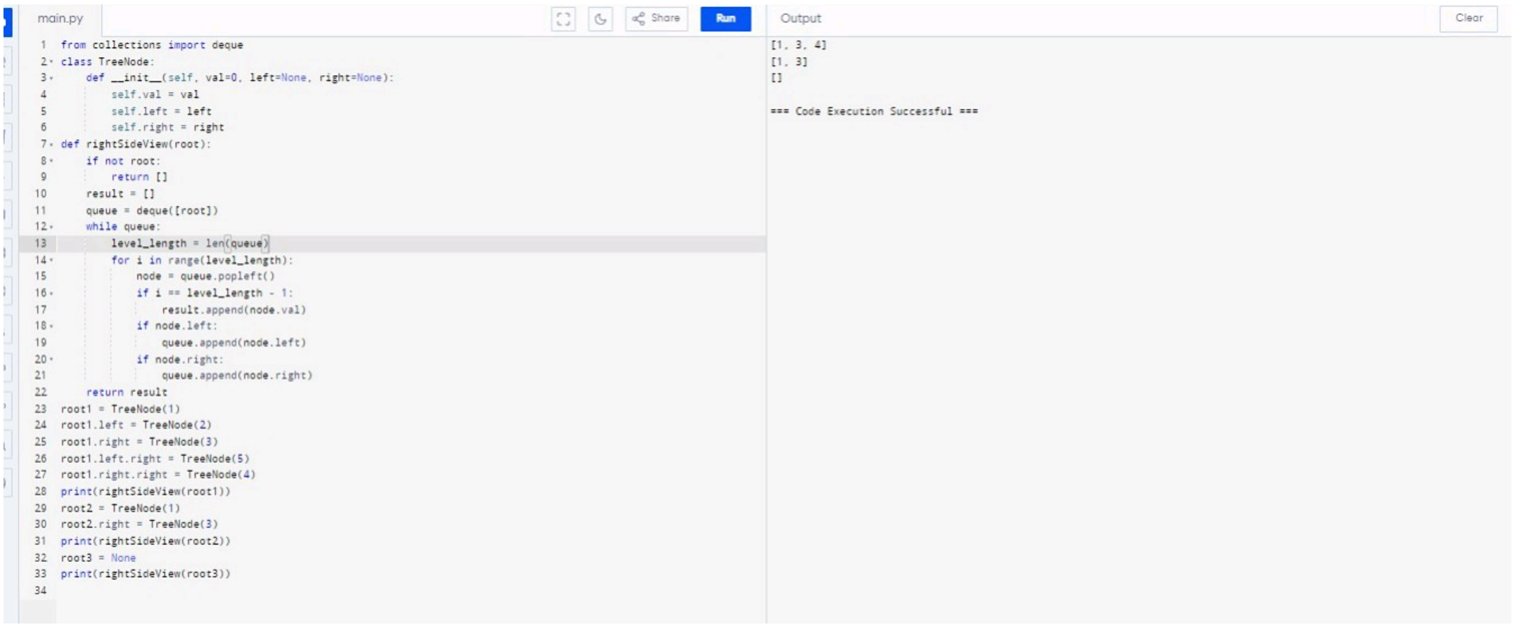
```python
def findMedianSortedArrays(nums1, nums2):
    nums = sorted(nums1 + nums2)
    n = len(nums)
    if n % 2 == 0:
        return (nums[n // 2 - 1] + nums[n // 2]) / 2
    else:
        return nums[n // 2]

nums1 = [1, 3]
nums2 = [2]
print(findMedianSortedArrays(nums1, nums2))

nums1 = [1, 2]
nums2 = [3, 4]
print(findMedianSortedArrays(nums1, nums2))
```

```
2
2.5

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



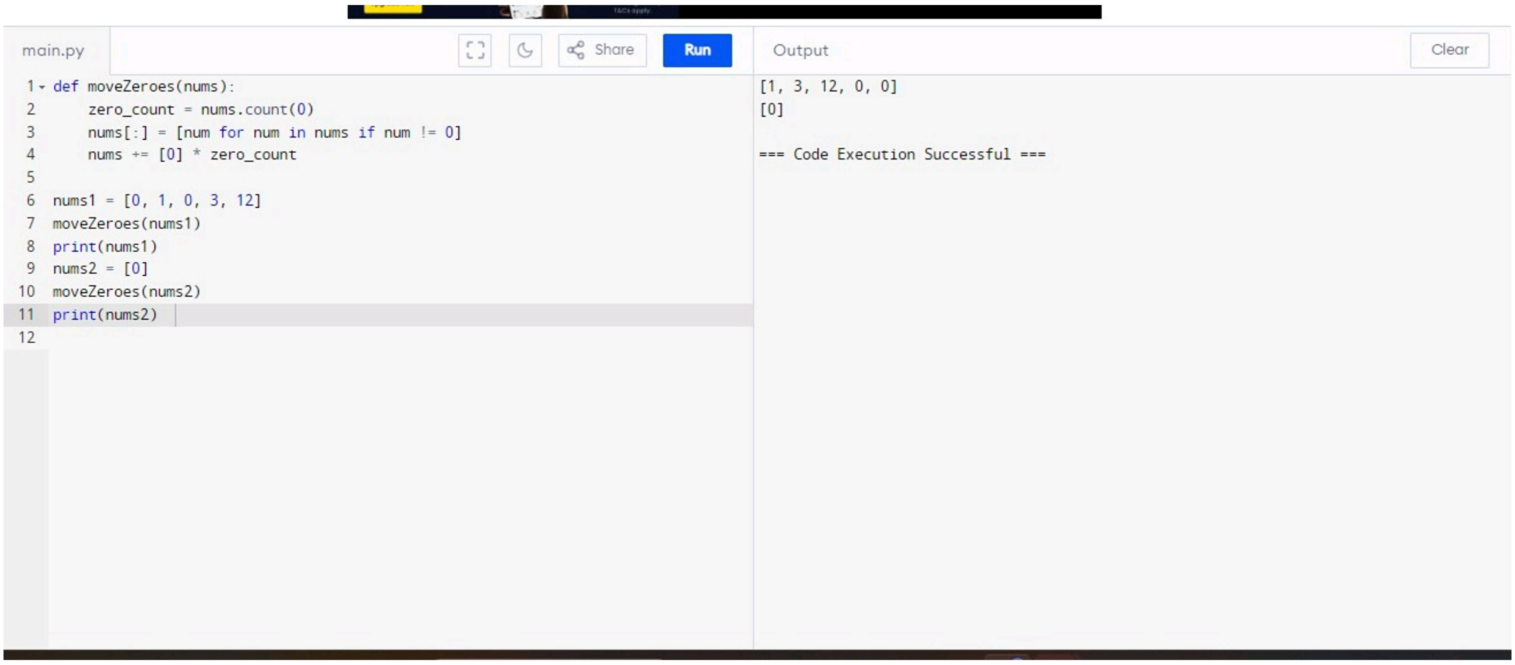
```python
def divide(dividend: int, divisor: int) -> int:
    if dividend == -2**31 and divisor == -1:
        return 2**31 - 1

    sign = -1 if (dividend < 0) ^ (divisor < 0) else 1

    dividend, divisor = abs(dividend), abs(divisor)

    quotient = 0
    while dividend >= divisor:
        dividend -= divisor
        quotient += 1

    if quotient > 2**31 - 1:
        return 2**31 - 1
    elif quotient < -2**31:
        return -2**31
    else:
        return sign * quotient
print(divide(10, 3))
print(divide(7, -3))
print(divide(-7, 3))
print(divide(-10, -3))
print(divide(-2**31, 1))
print(divide(2**31 - 1, 1))
```

```
3
-2
-2
3
```



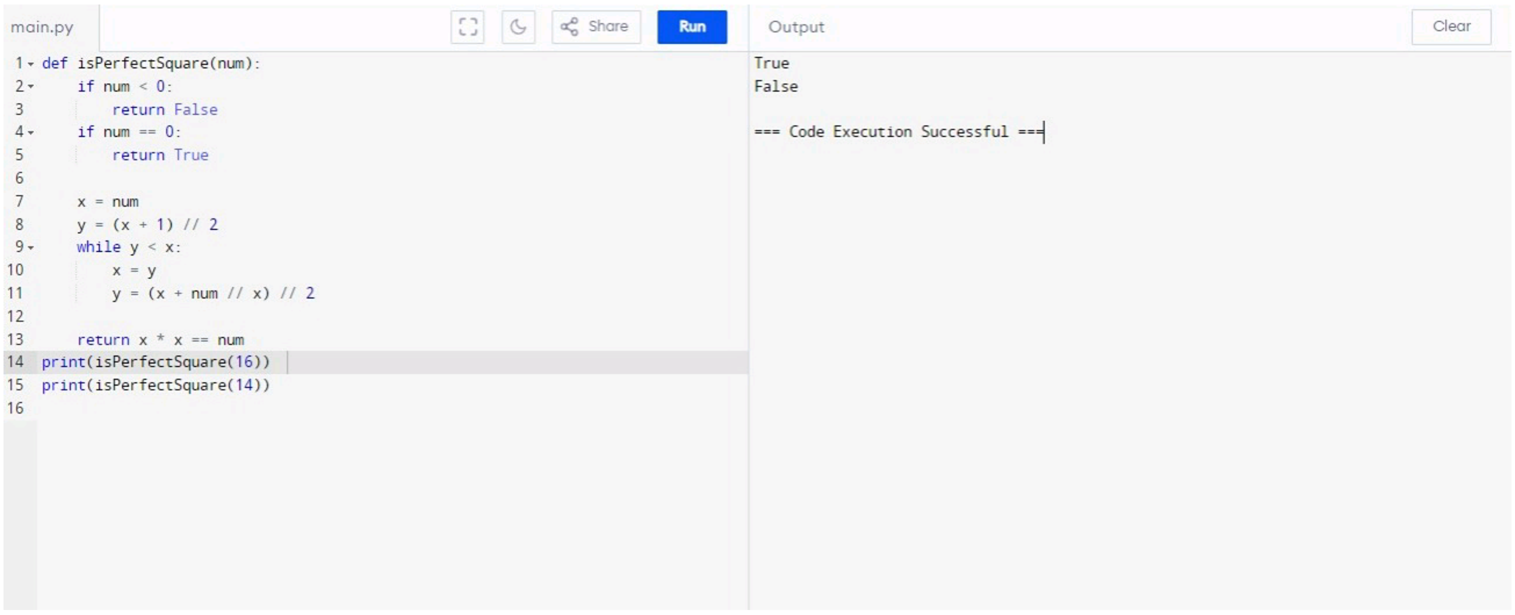
```python
from collections import deque
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def rightSideView(root):
    if not root:
        return []
    result = []
    queue = deque([root])
    while queue:
        level_length = len(queue)
        for i in range(level_length):
            node = queue.popleft()
            if i == level_length - 1:
                result.append(node.val)
            if node.left:
                queue.append(node.left)
            if node.right:
                queue.append(node.right)
    return result
root1 = TreeNode(1)
root1.left = TreeNode(2)
root1.right = TreeNode(3)
root1.left.right = TreeNode(5)
root1.right.right = TreeNode(4)
print(rightSideView(root1))
root2 = TreeNode(1)
root2.right = TreeNode(3)
print(rightSideView(root2))
root3 = None
print(rightSideView(root3))
```

```
[1, 3, 4]
[1, 3]
[]

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



```python
def moveZeroes(nums):
    zero_count = nums.count(0)
    nums[:] = [num for num in nums if num != 0]
    nums += [0] * zero_count

nums1 = [0, 1, 0, 3, 12]
moveZeroes(nums1)
print(nums1)
nums2 = [0]
moveZeroes(nums2)
print(nums2)
```

```
[1, 3, 12, 0, 0]
[0]

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



```python
def isPerfectSquare(num):
    if num < 0:
        return False
    if num == 0:
        return True

    x = num
    y = (x + 1) // 2
    while y < x:
        x = y
        y = (x + num // x) // 2

    return x * x == num
print(isPerfectSquare(16))
print(isPerfectSquare(14))
```

```
True
False

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