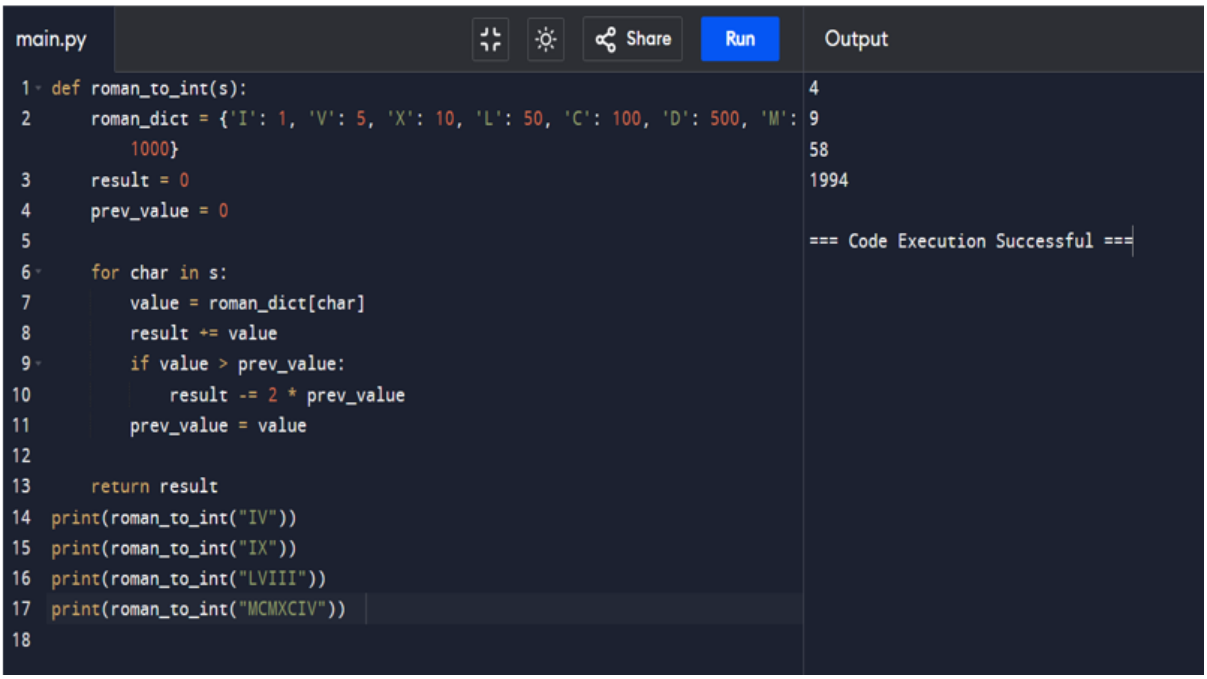
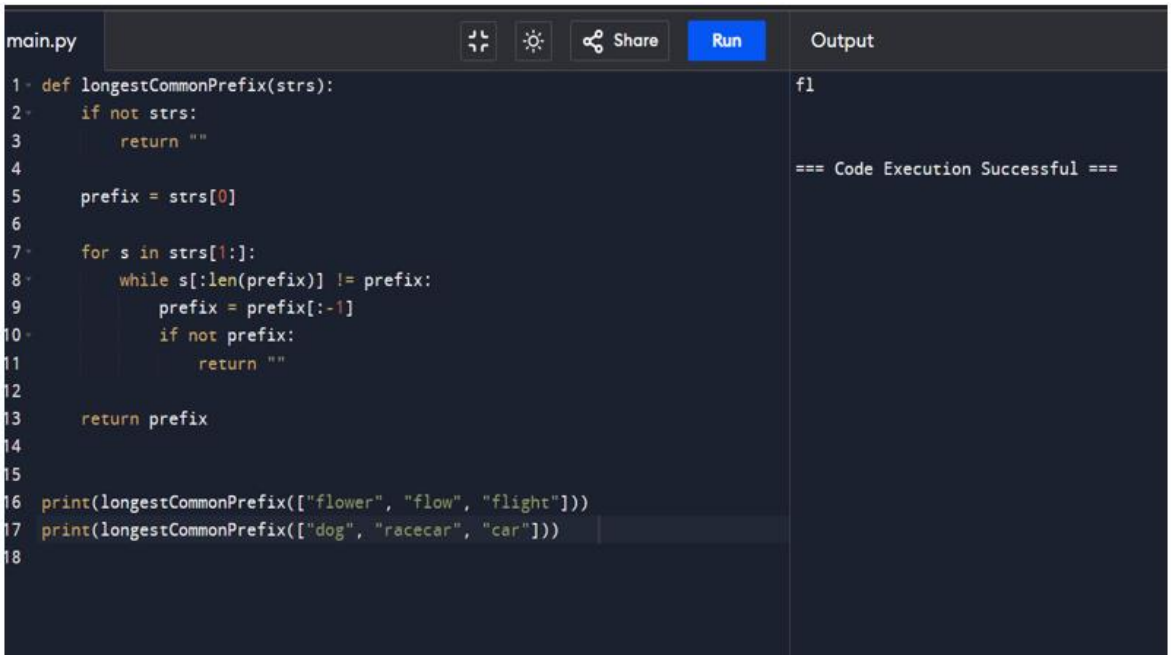
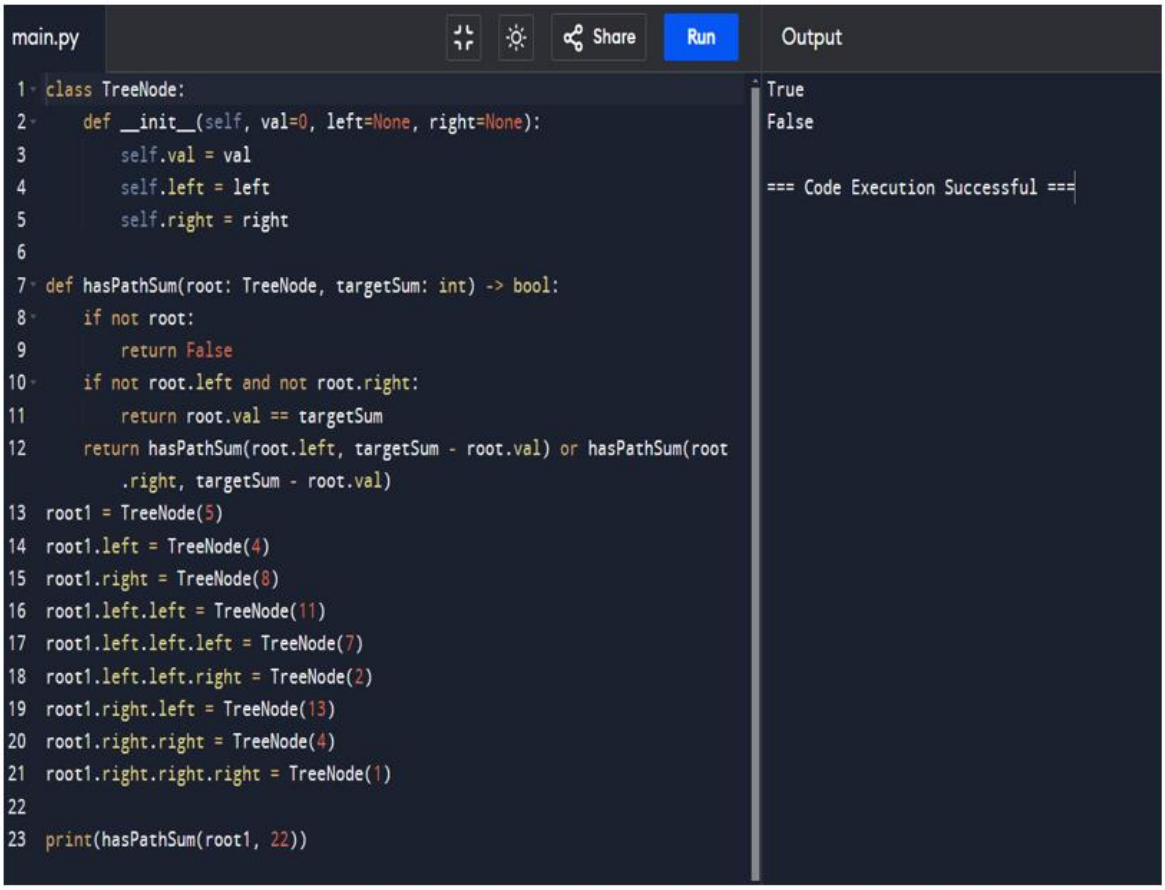
1.Converting Roman Numbers to integers?



2.



3.



4. Given the root of a binary tree and an integer of targetsum return true if the tree has a root to leaf such that adding up all the values

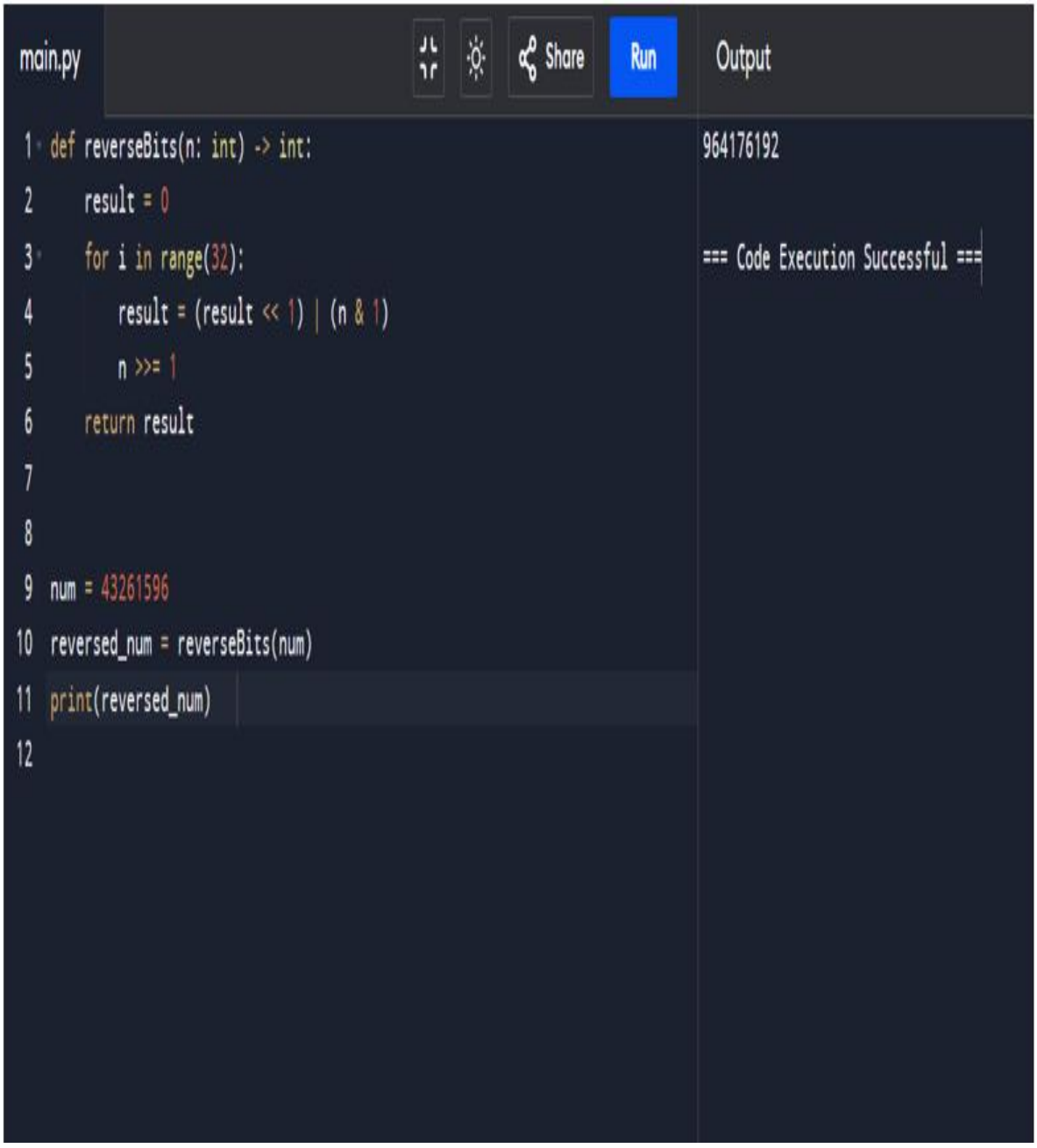
main.py

Share Run Output

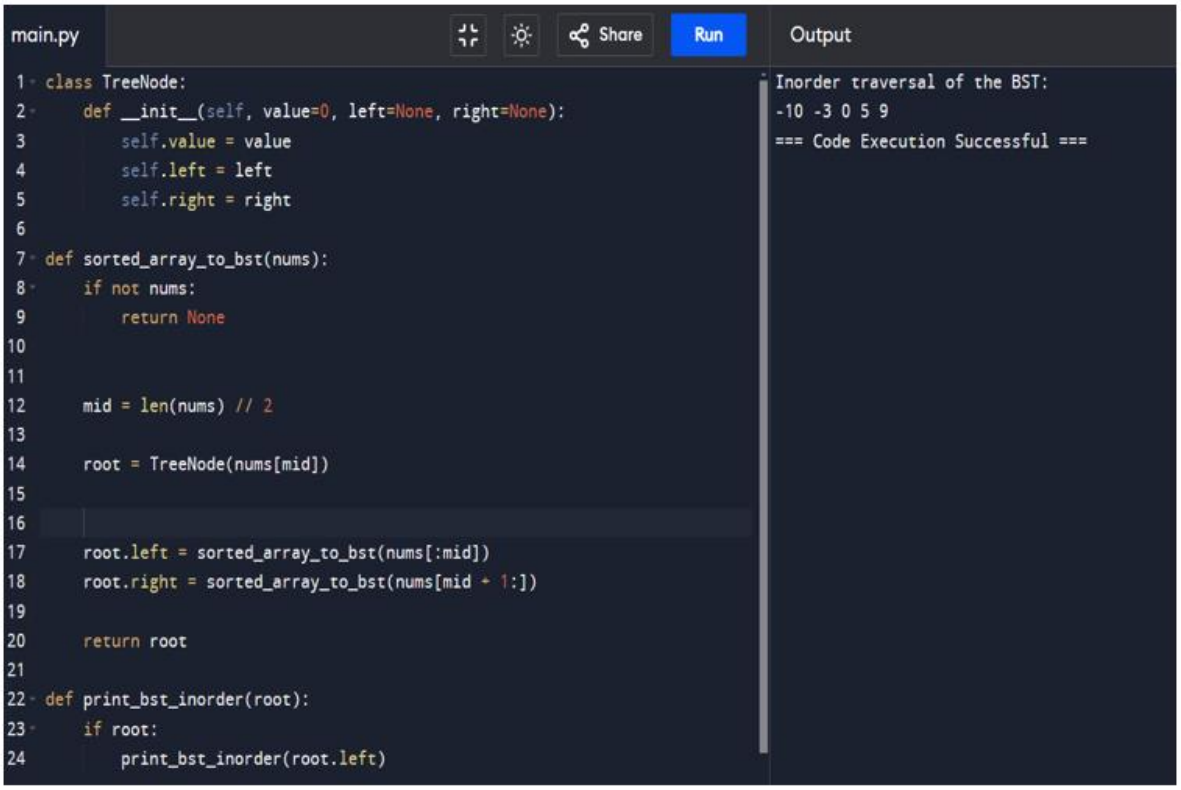
```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def hasPathSum(root: TreeNode, targetSum: int) -> bool:
    if not root:
        return False
    if not root.left and not root.right:
        return root.val == targetSum
    return hasPathSum(root.left, targetSum - root.val) or hasPathSum(root
        .right, targetSum - root.val)
root1 = TreeNode(5)
root1.left = TreeNode(4)
root1.right = TreeNode(8)
root1.left.left = TreeNode(11)
root1.left.left.left = TreeNode(7)
root1.left.left.right = TreeNode(2)
root1.right.left = TreeNode(13)
root1.right.right = TreeNode(4)
root1.right.right.right = TreeNode(1)

print(hasPathSum(root1, 22))
```

```
True
False

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

5.bit reversing?



6. convert sorted array to binary search tree?



7. given a binary tree, determine if it is height-balanced?

main.py | Share | Run | Output

```python
class TreeNode:
    def __init__(self, value=0, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right

def is_balanced(root):
    def check_height(node):
        if not node:
            return 0

        left_height = check_height(node.left)
        if left_height == -1:
            return -1

        right_height = check_height(node.right)
        if right_height == -1:
            return -1


        if abs(left_height - right_height) > 1:
            return -1

        return max(left_height, right_height) + 1
```

```
Is the tree balanced? True

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

8. Climbing stairs?

main.py | Share | Run | Output

```python
def climb_stairs(n):
    if n <= 1:
        return 1


    dp = [0] * (n + 1)


    dp[0] = 1
    dp[1] = 1


    for i in range(2, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]

    return dp[n]


n = 5
print("Number of ways to climb the stairs:", climb_stairs(n))
```

```
Number of ways to climb the stairs: 8

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

9. best time to buy and sell stock?

main.py

Share Run Output

```python
def max_profit(prices):
    if not prices or len(prices) < 2:
        return 0

    min_price = float('inf')
    max_profit = 0

    for price in prices:

        if price < min_price:
            min_price = price


        profit = price - min_price


        if profit > max_profit:
            max_profit = profit

    return max_profit


prices = [7, 1, 5, 3, 6, 4]
print("Maximum profit:", max_profit(prices))
```

```
Maximum profit: 5

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

10.sum of two binary strings?

main.py

Share Run Output

```python
def add_binary(a: str, b: str) -> str:

    int_a = int(a, 2)
    int_b = int(b, 2)


    sum_int = int_a + int_b


    return bin(sum_int)[2:]


binary_str1 = "1010"
binary_str2 = "1101"
print("Sum of binary strings:", add_binary(binary_str1, binary_str2))

```

```
Sum of binary strings: 10111

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