ASSIGNMENT-6

NAME: D. Guna vardhan

REG-NO: 192372041

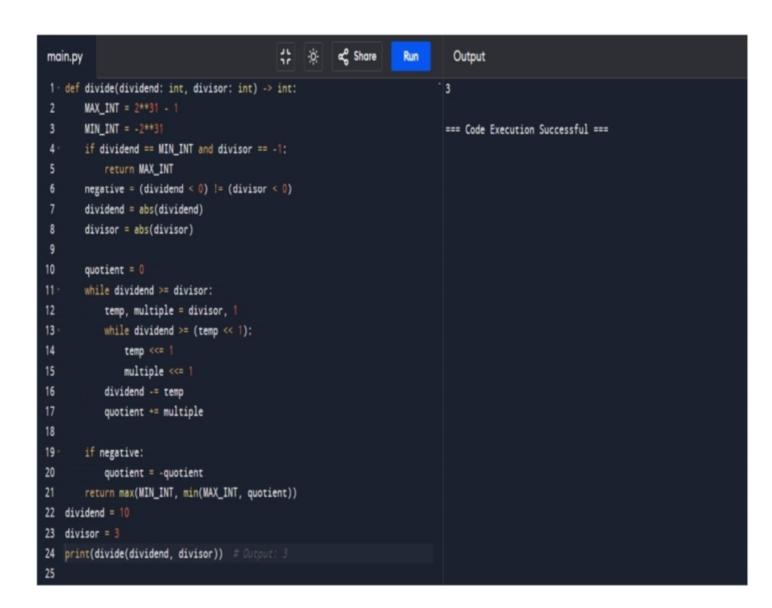
SUBJECT: Python

CODE: CSA0898

1. Given two sorted arrays nums 1 and nums 2 of size m and n respectively, return the median of the two sorted arrays. The overall run time complexity should be $O(\log (m+n))$. Example 1: Input: nums 1 = [1,3], nums 2 = [2] Output: 2.00000

```
ℴ℃ Share
                                                                                  Output
main.py
                                                                        Run
 1 def findMedianSortedArrays(nums1, nums2):
        nums = sorted(nums1 + nums2)
3
       n = len(nums)
                                                                                 === Code Execution Successful ===
        if n % 2 == 0:
            return (nums[n // 2 - 1] + nums[n // 2]) / 2
            return nums[n // 2]
9 \text{ nums1} = [1, 3]
10 \text{ nums2} = [2]
   print(findMedianSortedArrays(nums1, nums2))
12
```

2.Given two integers dividend and divisor, divide two integers without using multiplication, division, and mod operator. The integer division should truncate toward zero, which means losing its fractional part. For example, 8.345 would be truncated to 8, and -2.7335 would be truncated to -2. Return the quotient after dividing dividend by divisor. Note: Assume we are dealing with an environment that could only store integers within the 32-bit signed integer range: [-231, 231 - 1]. For this problem, if the quotient is strictly greater than 231 - 1, then return 231 - 1, and if the quotient is strictly less than -231, then return -231.



3. Given the root of a binary tree, imagine yourself standing on the right side of it, return the values of the nodes you can see ordered from top to bottom.

```
Programiz Python Online Compiler
                                                             ∞ Share
                                                 ₹
                                                                          Run
                                                                                     Output
main.py
    from collections import deque
                                                                                   [1, 3, 4]
    class TreeNode:
        def __init__(self, val=0, left=None, right=None):
                                                                                   === Code Execution Successful ===
            self.val = val
            self.left = left
 5
 6
            self.right = right
    def rightSideView(root: TreeNode):
 9
        if not root:
10
            return []
       right_view, queue = [], deque([root])
13
14
        while queue:
15
            level_length = len(queue)
16
            for i in range(level_length):
17
                node = queue.popleft()
18
19
20
21
22
                if i == level_length - 1:
                    right_view.append(node.val)
                if node.left:
                    queue.append(node.left)
                if node.right:
23
                    queue.append(node.right)
24
```

4. Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the non-zero elements. Note that you must do this in-place without making a copy of the array. Example 1: Input: nums = [0,1,0,3,12] Output: [1,3,12,0,0]

```
∞ Share
                                                                                 Output
main.py
                                                                       Run
                                                                               [1, 3, 12, 0, 0]
1 def moveZeroes(nums):
       n = len(nums)
       last_non_zero_found_at = 0
                                                                                === Code Execution Successful ===
3
       for i in range(n):
6
           if nums[i] != 0:
               nums[last_non_zero_found_at], nums[i] = nums[i],
                   nums[last_non_zero_found_at]
               last_non_zero_found_at += 1
  nums = [0, 1, 0, 3, 12]
12 moveZeroes(nums)
  print(nums)
14
```

5. . Given a positive integer num, return true if num is a perfect square or false otherwise. A perfect square is an integer that is the square of an integer. In other words, it is the product of some integer with itself. You must not use any built-in library function, such as sqrt. Example 1: Input: num = 16 Output: true

```
∞ Share
main.py
                                                                        Run
                                                                                  Output
1 def isPerfectSquare(num):
                                                                                 True
       if num < 0:
3
                                                                                 === Code Execution Successful ===
          return False
       if num == 0:
5
          return True
6
       x = num
       y = (x + 1) // 2
8
       while y < x:
10
           y = (x + num // x) // 2
       return x * x == num
12
   num = 16
  print(isPerfectSquare(num))
```