ASSIGMNENT→6

A. Harsha Mukundha
192324070

Questions:

1. Given two sorted arrays nums1 and nums2 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: nums1 = [1,3], nums2 = [2]

Output: 2.00000

Explanation: merged array = [1,2,3] and median is 2.

Example 2:

Input: nums1 = [1,2], nums2 = [3,4]

Output: 2.50000

Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

Constraints:

nums1.length == m

nums2.length == n

0 <= m <= 1000

0 <= n <= 1000

1 <= m + n <= 2000

-106 <= nums1[i], nums2[i] <= 106

Run Save

```python
def findMedianSortedArrays(nums1, nums2):
    if len(nums1) > len(nums2):
        nums1, nums2 = nums2, nums1
    m, n = len(nums1), len(nums2)
    imin, imax, half_len = 0, m, (m + n + 1) // 2
    while imin <= imax:
        i = (imin + imax) // 2
        j = half_len - i
        if i < m and nums1[i] < nums2[j-1]:
            imin = i + 1
        elif i > 0 and nums1[i-1] > nums2[j]:
            imax = i - 1
        else:
            if i == 0:
                max_of_left = nums2[j-1]
            elif j == 0:
                max_of_left = nums1[i-1]
            else:
                max_of_left = max(nums1[i-1], nums2[j-1])
            if (m + n) % 2 == 1:
                return max_of_left
            if i == m:
                min_of_right = nums2[j]
            elif j == n:
                min_of_right = nums1[i]
            else:
```

```
1 3
2
```

```
The median is: 2
```

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.

Example 1:

Input: dividend = 10, divisor = 3

Output: 3

Explanation: 10/3 = 3.33333.. which is truncated to 3.

Example 2:

Input: dividend = 7, divisor = -3

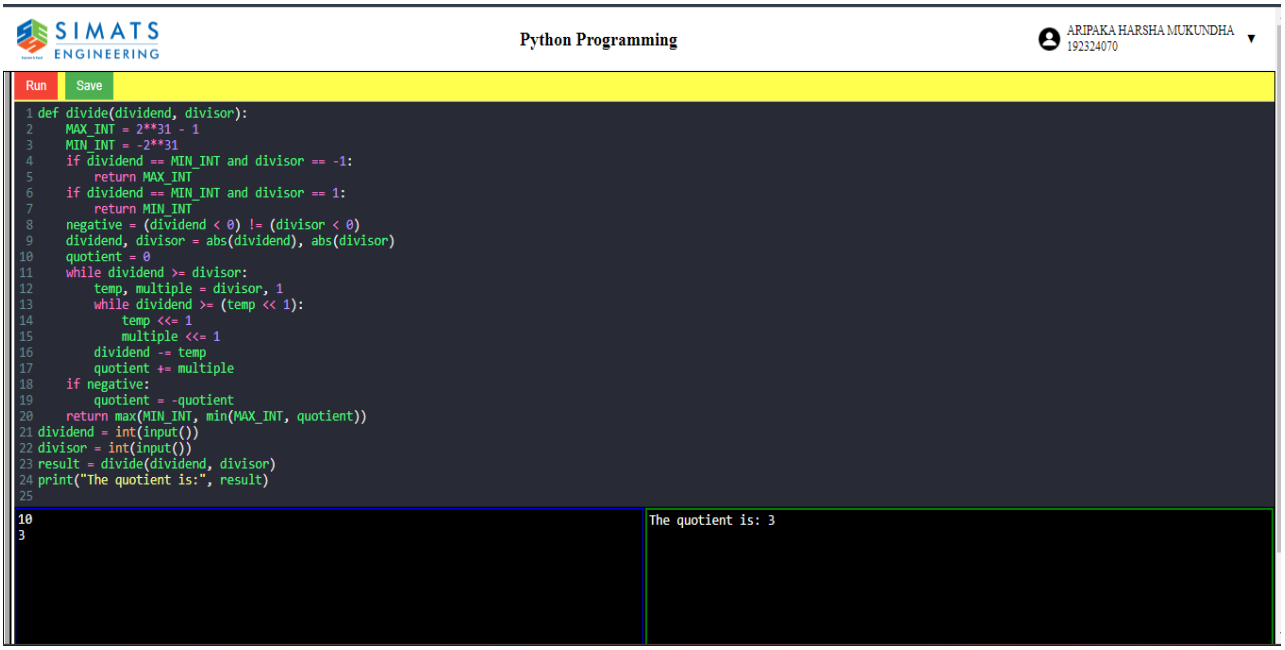
Output: -2

Explanation: 7/-3 = -2.33333.. which is truncated to -2.

Constraints:

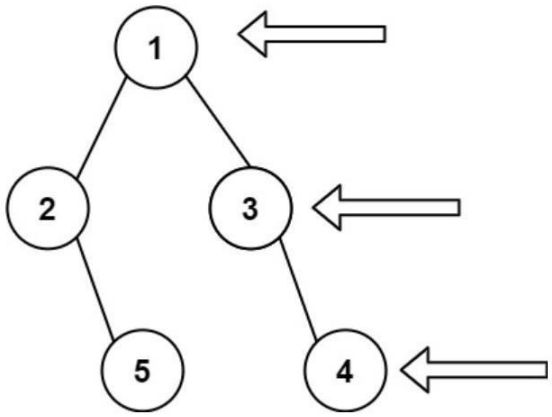
-231 <= dividend, divisor <= 231 - 1

divisor != 0



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.

Example 1:



Input: root = [1, 2, 3, null, 5, null, 4]

Output: [1,3,4]

Example 2:

Input: root = [1,null,3]

Output: [1,3]

SIMATS ENGINEERING

Python Programming

ARIPAKA HARSHA MUKUNDHA
192324070

Run Save

```
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 []
    right_view = []
    queue = [root]
    while queue:
        level_length = len(queue)
        for i in range(level_length):
            node = queue.pop(0)
            if i == level_length - 1:
                right_view.append(node.val)
            if node.left:
                queue.append(node.left)
            if node.right:
                queue.append(node.right)
    return right_view
def insertLevelOrder(arr, root, i, n):
    if i < n:
        temp = TreeNode(arr[i]) if arr[i] is not None else None
        root = temp
```

```
1, 2, 3, 5, 4
```

```
The right side view of the binary tree is: [1, 3, 4]
```

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]

Example 2:

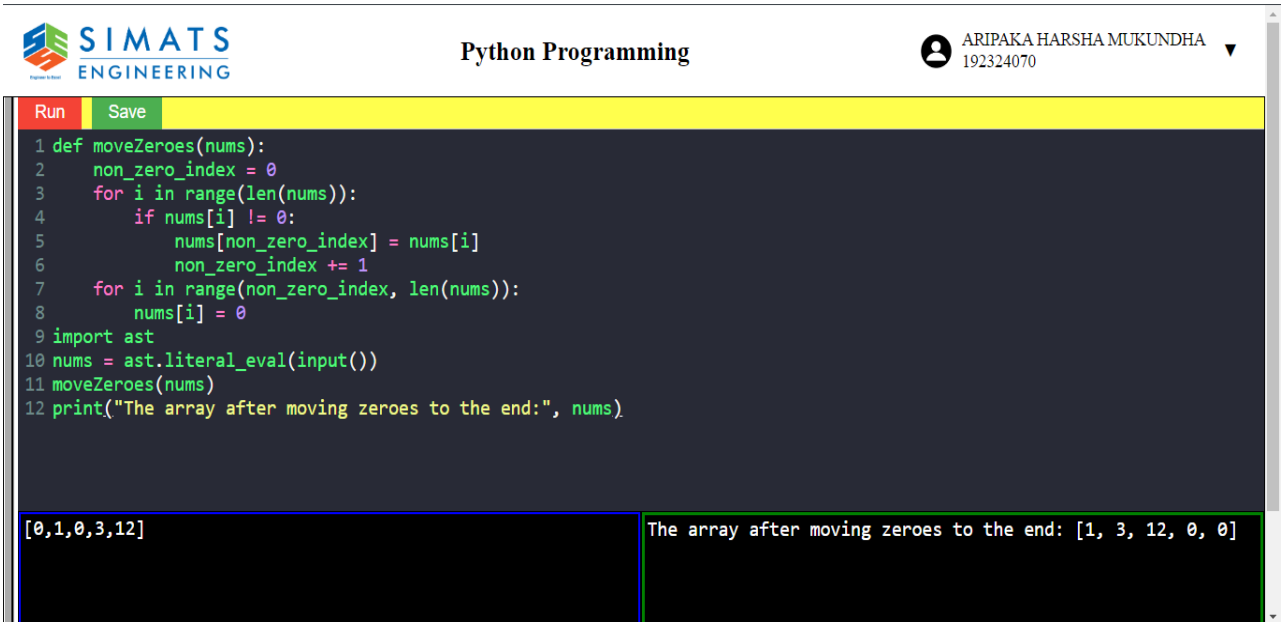
Input: nums = [0]

Output: [0]

Constraints:

1 <= nums.length <= 104

-231 <= nums[i] <= 231 – 1



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

Explanation: We return true because 4 * 4 = 16 and 4 is an integer.

Example 2:

Input: num e= 14

Output: false