

## ASSIGNMENT-6

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`  $0 \leq m \leq$

1000

$0 \leq n \leq 1000$

$1 \leq m + n \leq 2000$

$-106 \leq \text{nums1}[i], \text{nums2}[i] \leq 106$

```
6-1.py - C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-1.py (3.12.4)
File Edit Format Run Options Window Help
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))

IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-1.py ====
2
2.5
>>>
```

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:  $[-2^{31}, 2^{31} - 1]$ . For this problem, if the quotient is strictly greater than  $2^{31} - 1$ , then return  $2^{31} - 1$ , and if the quotient is strictly less than  $-2^{31}$ , then return  $-2^{31}$ .

Example 1:

Input: dividend = 10, divisor = 3

Output: 3

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

Example 2:

Input: dividend = 7, divisor = -3

Output: -2

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

Constraints:

$-2^{31} \leq \text{dividend}, \text{divisor} \leq 2^{31} - 1$

divisor  $\neq 0$

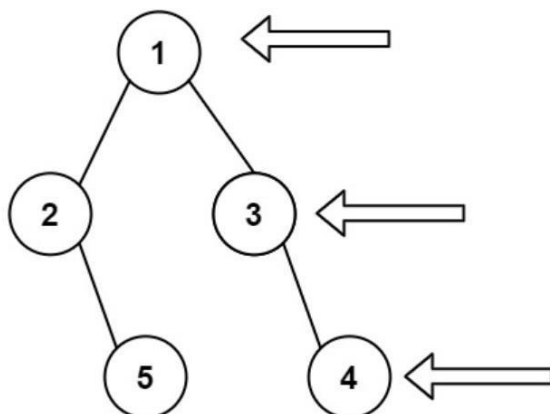
```
6-2.py - C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-2.py (3.12.4)
File Edit Format Run Options Window Help
def divide(dividend, divisor):
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31
    if dividend == 0:
        return 0
    if dividend == INT_MIN and divisor == -1:
        return INT_MAX
    negative = (dividend < 0) != (divisor < 0)
    dividend, divisor = abs(dividend), abs(divisor)
    quotient = 0
    while dividend >= divisor:
        temp, i = divisor, 1
        while dividend >= temp:
            dividend -= temp
            quotient += i
            i <<= 1
            temp <<= 1
    return -quotient if negative else quotient

print(divide(10, 3))

IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
- RESTART: C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-2.py
>>>
3
>>>
```

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]

Example 3:

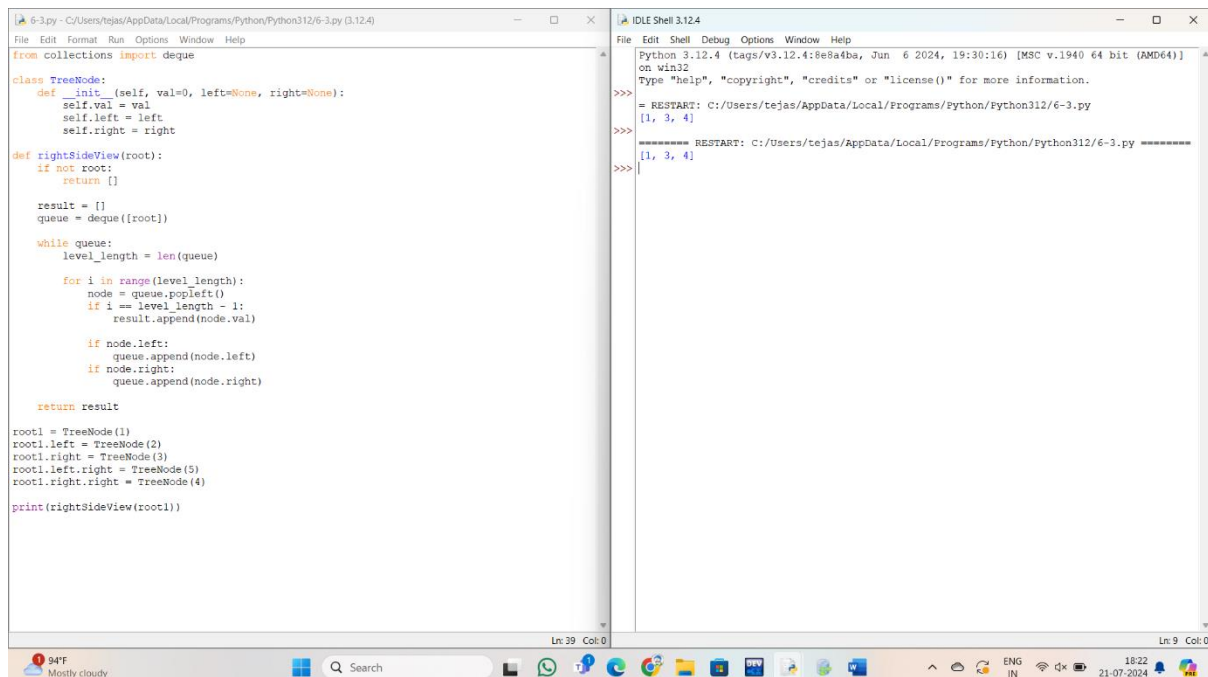
Input: root = []

Output: []

Constraints:

The number of nodes in the tree is in the range  $[0, 100]$ .

$-100 \leq \text{Node.val} \leq 100$



The screenshot shows a Python IDE with two windows. The left window displays a Python script for a binary tree. The right window shows the execution output.

```
6-3.py - C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-3.py (3.12.4)
File Edit Format Run Options Window Help
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))
```

```
IDLE Shell 3.12.4
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-3.py
[1, 3, 4]
>>>
===== RESTART: C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-3.py =====
[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:

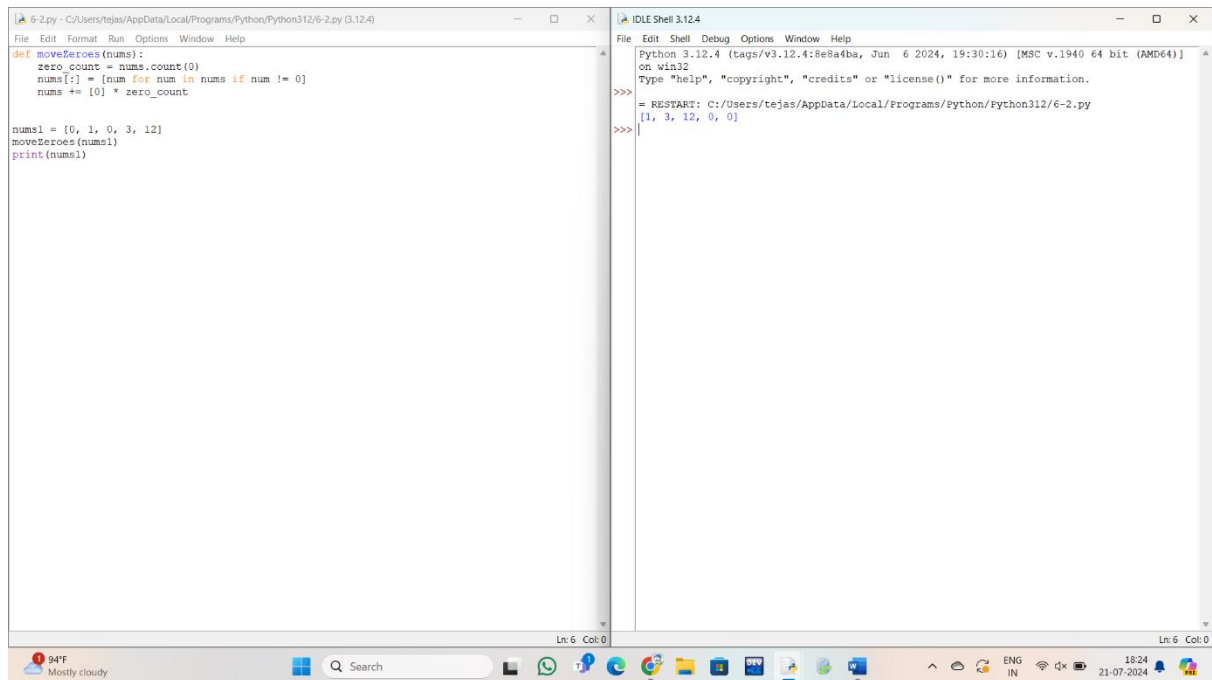
Input: `nums = [0]`

Output: `[0]`

Constraints:

$1 \leq \text{nums.length} \leq 104$

$-231 \leq \text{nums}[i] \leq 231 - 1$



The screenshot shows an IDE with two windows. The left window, titled '6-2.py', contains the following Python code:

```
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)
```

The right window, titled 'IDLE Shell 3.12.4', shows the execution output:

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/tejas/AppData/Local/Programs/Python/Python312/6-2.py
[1, 3, 12, 0, 0]
>>>
```

The taskbar at the bottom shows the system clock as 18:24 on 21-07-2024.

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 = 14

Output: false

Explanation: We return false because  $3.742 * 3.742 = 14$  and 3.742 is not an integer.

Constraints:

2 ≤ num ≤ 231 –

