LAB PROGRAMS 1-5

1. Given an integer array num sorted in non-decreasing order. You can perform the following operation any number of times: Choose two indices, i and j, where nums[i] < nums[j]. Then, remove the elements at indices i and j from nums. The remaining elements retain their original order, and the array is re□indexed. Return the minimum length of nums after applying the operation zero or more times.
   Example 1:
   Input: nums = [1,2,3,4]
   Output: 0
   Constraints: 1 <= nums.length <= 105 1 <= nums[i] <= 109 nums is sorted in non-decreasing order.

PROGRAM:

```
def min_length_of_array(nums):
    stack = []
    for num in nums:
        if stack and num < stack[-1]:
            stack.pop()
        else:
            stack.append(num)
    return len(stack)

nums = [1, 2, 3, 4]
print(min_length_of_array(nums))
```

OUTPUT:

```
==== RESTART
4
```

2. Given an integer array nums where the elements are sorted in ascending order, convert it to a height-balanced binary search tree.

   Example 1:

   Input: nums = [-10,-3,0,5,9]

   Output: [0,-3,9,-10,null,5]

Explanation: [0,-10,5,null,-3,null,9] is also accepted

PROGRAM:

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

def sortedArrayToBST(nums):
    if not nums:
        return None

    mid = len(nums) // 2
    root = TreeNode(nums[mid])
    root.left = sortedArrayToBST(nums[:mid])
    root.right = sortedArrayToBST(nums[mid + 1:])

    return root

nums = [-10, -3, 0, 5, 9]
result = sortedArrayToBST(nums)
```

OUTPUT:

3. Given an array of string words, return all strings in words that is a substring of another word. You can return the answer in any order. A substring is a contiguous sequence of characters within a string.
   Example 1:
   Input: words = ["mass","as","hero","superhero"]
   Output: ["as","hero"]
   Explanation: "as" is substring of "mass" and "hero" is substring of "superhero". ["hero","as"] is also a valid answer.

PROGRAM:

```
def stringMatching(words):
    return [word for word in words if any(other_word != word and other_word.find(word) != -1 for other_word in words)]

words = ["mass", "as", "hero", "superhero"]
print(stringMatching(words))
```

OUTPUT:

```
==== RESTART: C:/Us
['as', 'hero']
```

4. Given an integer array nums, reorder it such that nums[0] < nums[1] > nums[2] < nums[3].... You may assume the input array always has a valid answer.
   Example 1:
   Input: nums = [1,5,1,1,6,4]
   Output: [1,6,1,5,1,4]
   Explanation: [1,4,1,5,1,6] is also accepted.
   Example 2:
   Input: nums = [1,3,2,2,3,1]
   Output: [2,3,1,3,1,2]

PROGRAM:

```
def wiggleSort(nums):
    nums.sort()
    n = len(nums)

    mid = (n + 1) // 2
    left = nums[:mid]
    right = nums[mid:]

    left.reverse()
    right.reverse()

    nums[::2] = left
    nums[1::2] = right

nums1 = [1, 5, 1, 1, 6, 4]
wiggleSort(nums1)
print(nums1)
```

5. Given an m x n binary matrix mat, return the distance of the nearest 0 for each cell. The distance between two adjacent cells is 1.
   Example 1:
   Input: mat = [[0,0,0],[0,1,0],[0,0,0]]
   Output: [[0,0,0],[0,1,0],[0,0,0]]
   Example 2:
   Input: mat = [[0,0,0],[0,1,0],[1,1,1]]
   Output: [[0,0,0],[0,1,0],[1,2,1]]

PROGRAM:

```
from collections import deque

def updateMatrix(mat):
    rows, cols = len(mat), len(mat[0])
    queue = deque()

    for i in range(rows):
        for j in range(cols):
            if mat[i][j] == 0:
                queue.append((i, j))
            else:
                mat[i][j] = float('inf')

    directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]

    while queue:
        cell = queue.popleft()
        for d in directions:
            new_i, new_j = cell[0] + d[0], cell[1] + d[1]
            if 0 <= new_i < rows and 0 <= new_j < cols and mat[new_i][new_j] > mat[cell[0]][cell[1]] + 1:
                mat[new_i][new_j] = mat[cell[0]][cell[1]] + 1
                queue.append((new_i, new_j))

    return mat

mat1 = [[0, 0, 0], [0, 1, 0], [0, 0, 0]]
mat2 = [[0, 0, 0], [0, 1, 0], [1, 1, 1]]

print(updateMatrix(mat1))
print(updateMatrix(mat2))
```

OUTPUT:

```
==== RESTART: C:/Users/mahum/AppData/
[[0, 0, 0], [0, 1, 0], [0, 0, 0]]
[[0, 0, 0], [0, 1, 0], [1, 2, 1]]
```