Assignment

Easy 1

Given a string `s` consisting of words and spaces, return *the length of the **last** word in the string.*
A **word** is a maximal
substring consisting of non-space characters only.
Example 1:
Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.
Example 2:
Input: s = " fly me to the moon "
Output: 4
Explanation: The last word is "moon" with length 4.
Example 3:
Input: s = "luffy is still joyboy"
Output: 6
Explanation: The last word is "joyboy" with length 6.
Constraints:

- `1 <= s.length <= 104`
- `s` consists of only English letters and spaces `' '`.
- There will be at least one word in `s`.

Code:

```
def length_of_last_word(s):
    words=s.split()
    if not words:
        return 0
    return len(words[-1])
input_string = input()
output_length = length_of_last_word(input_string)
print(f"Input: {input_string}\nOutput: {output_length}")
```

Medium 2

Given an integer array of size `n`, find all elements that appear more than `⌊ n/3 ⌋` times.
Example 1:
Input: nums = [3,2,3]
Output: [3]
Example 2:
Input: nums = [1]
Output: [1]
Example 3:
Input: nums = [1,2]
Output: [1,2]
Constraints:

- `1 <= nums.length <= 5 * 104`
- `-109 <= nums[i] <= 109`

Code:

```
def majority_elements(nums):
    if not nums:
        return []

count1, count2, candidate1, candidate2 = 0, 0, 0, 1

for num in nums:
    if num == candidate1:
        count1 += 1
    elif num == candidate2:
        count2 += 1
    elif count1 == 0:
        candidate1, count1 = num, 1
    elif count2 == 0:
        candidate2, count2 = num, 1
    else:
        count1 -= 1
        count2 -= 1

count1, count2 = 0, 0
```

```
    for num in nums:
        if num == candidate1:
            count1 += 1
        elif num == candidate2:
            count2 += 1
result = []
    if count1 > len(nums) // 3:
        result.append(candidate1)
    if count2 > len(nums) // 3:
        result.append(candidate2)

    return result

# Example usage:
nums1 = [3, 2, 3]
nums2 = [1]
nums3 = [1, 2]

print(majority_elements(nums1))  # Output: [3]
print(majority_elements(nums2))  # Output: [1]
print(majority_elements(nums3))  # Output: [1, 2]
```

Hard 2

You are given a string `s`. You can convert `s` to a
palindrome by adding characters in front of it.
Return *the shortest palindrome you can find by performing this transformation*.
Example 1:
Input: s = "aacecaaa"
Output: "aaacecaaa"
Example 2:
Input: s = "abcd"
Output: "dcbabcd"
Constraints:

- `0 <= s.length <= 5 * 104`
- `s` consists of lowercase English letters only.

Code:

```
def shortestPalindrome(s):
    i = 0
    for j in range(len(s) - 1, -1, -1):
        if s[i] == s[j]:
            i += 1
    if i == len(s):
        return s
    else:
        return s[i:][::-1] + shortestPalindrome(s[:i]) + s[i:]


# Example usage:
print(shortestPalindrome("aacecaaa"))  # Output: "aaacecaaa"
print(shortestPalindrome("abcd"))      # Output: "dcbabcd"
```