EXPLANATION FOR THE CODES:

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.

SOLUTION :

```
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        return len(s.rstrip().split(' ')[-1])
```

EXPLANATION:

1. Class Definition:

```
class Solution:
```

The code defines a class named `Solution`. However, in this case, the class doesn't contain any instance variables or methods besides the `lengthOfLastWord` method.

2. lengthOfLastWord Method:

```
def lengthOfLastWord(self, s: str) -> int:
```

- `lengthOfLastWord` is a method within the `Solution` class.

- It takes in two parameters: `self` (implicitly passed in Python classes) and `s`, which is a string. It specifies that `s` must be a string.

- The method returns an integer, which represents the length of the last word in the input string `s`.

3. Code Logic:

```
return len(s.rstrip().split(' ')[-1])
```

- `s.rstrip()` is used to remove any trailing whitespace characters at the end of the string `s`.

- `split(' ')` is used to split the string into a list of substrings separated by spaces.

- `[-1]` is used to access the last element of the resulting list, which represents the last word in the string after splitting it by spaces.

- `len()` is then used to calculate the length of this last word.

4. Return Value:

- The method returns the length of the last word found in the input string `s`.

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

SOLUTION :

```
from collections import Counter

class Solution:

    def majorityElement(self, nums: List[int]) -> List[int]:
```

```
        count = Counter(nums)

        n = len(nums)

        return [key for key, value in count.items() if value > (n // 3)]
```

EXPLANATION:

1. `class Solution:`: This line declares a class named `Solution`. Classes are used to organize code into logical units, and in this case, it encapsulates a solution for finding the length of the last word in a string.

2. `def lengthOfLastWord(self, s: str) -> int:`: This line defines a method within the `Solution` class named `lengthOfLastWord`. This method takes in a string `s` as input and specifies that it will return an integer (`-> int`).

3. `return len(s.rstrip().split(' ')[-1])`: This line contains the logic to find the length of the last word in the input string.

   - `s.rstrip()`: `rstrip()` is a method that removes any trailing whitespace characters (spaces, tabs, etc.) from the right end of the string `s`.

   - `split(' ')`: `split(' ')` divides the string into a list of substrings using space (' ') as a delimiter. This splits the string into words wherever a space is encountered.

   - `[-1]`: After splitting the string into a list of words, `[-1]` is used to access the last element of the list (which represents the last word in the string).

   - `len(...)`: Finally, `len(...)` calculates the length of the last word obtained after the string is stripped of trailing spaces and split into words.

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.

SOLUTION:

```
class Solution:

    def shortestPalindrome(self, s: str) -> str:

        end = 0

        if(s == s[::-1]):

            return s

        for i in range(len(s)+1):

            if(s[:i]==s[:i][::-1]):

                end=i-1
```

```
        return (s[end+1:][::-1])+s
```

EXPLANATION:

1. `class Solution:`: This line declares a class named `Solution`. Classes are used to organize code into logical units.

2. `def shortestPalindrome(self, s: str) -> str:`: This line defines a method within the `Solution` class named `shortestPalindrome`. This method takes in a string `s` as input and specifies that it will return a string (`-> str`).

3. `end = 0`: Initializes a variable `end` to 0. This variable is used to store the index that determines where to split the string for creating the shortest palindrome.

4. `if(s == s[::-1]):`: Checks if the input string `s` is already a palindrome (a string that reads the same forward and backward). If `s` is a palindrome, it returns `s` as the shortest palindrome.

5. `for i in range(len(s)+1):`: This for loop iterates over the range from 0 to the length of the string `s` plus one.

6. `if(s[:i]==s[:i][::-1]):`: Inside the loop, it checks if the substring of `s` from index 0 to `i` is a palindrome. If it is, it updates the `end` variable to `i - 1`.

7. `return (s[end+1:][::-1])+s`: After the loop completes, this line constructs the shortest palindrome using the updated `end` index. It takes the substring of `s` from `end + 1` to the end, reverses it (`[::-1]`), and concatenates it with the original string `s`.