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.

```
Def length_of_last_word(s):
 # Split the string into words
 Words = s.split()

 # Check if there are any words
 If not words:
    Return 0

 # Return the length of the last word
 Return len(words[-1])

# Example usage:
Input_string = “Hello World”
Output_length = length_of_last_word(input_string)
Print(output_length)
```

2)Given an integer array of size n, find all elements that appear more than ⌊ n/3 ⌋ times.

```
 Def majority_elements(nums):
   If not nums:
      Return []

   # Initialize candidates and counters
   Candidate1, count1 = None, 0
Candidate2, count2 = None, 0

   # Voting process
```

```
    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

    # Count occurrences of candidates
    Count1, count2 = 0, 0
For num in nums:
        If num == candidate1:
            Count1 += 1
        Elif num == candidate2:
            Count2 += 1

    # Check if candidates appear more than n/3 times
    N = len(nums)
    Result = []
    If count1 > n // 3:
Result.append(candidate1)     If
count2 > n // 3:
        Result.append(candidate2)
```

```
    Return result

# Example usage:
Nums_input = [3, 2, 3]
Output_result = majority_elements(nums_input)
Print(output_result)
```

3)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.

```
    def shortest_palindrome(s):
    If not s:
        Return “”

    # Helper function to check if a string is a palindrome
    Def is_palindrome(string):
        Return string == string[::-1]

    N = len(s)

    # Find the longest palindrome prefix
    I = n
    While I > 0 and not is_palindrome(s[:i]):
        I -= 1

    # Construct the shortest palindrome
    Return s[i:][::-1] + s
```

```
# Example usage:
Input_string1 = “aacecaaa”
Output_result1 = shortest_palindrome(input_string1)
Print(output_result1)

Input_string2 = “abcd”
Output_result2 = shortest_palindrome(input_string2)
Print(output_result2)
```