Love Local 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.

Code:

length.py - C:/Users/034354744/Documents/Deesha/length.py (3.10.11)

File Edit Format Run Options Window Help

```python
def length_of_last_word(s):
    words = s.split()
    if not words:
        return 0
    return len(words[-1])

print(length_of_last_word(input()))
```

Output:

IDLE Shell 3.10.11

File Edit Shell Debug Options Window Help

```
Python 3.10.11 (tags/v3.10.11:7d4c
 (AMD64)] on win32
Type "help", "copyright", "credits
>>>
============ RESTART: C:\Users\034
Hello World
5
>>>
============ RESTART: C:\Users\034
  fly me  to  the moon
4
>>>
============ RESTART: C:\Users\034
luffy is still joyboy
6
>>>
```

Logic and Algorithm of the code:

1.Function Definition:

The function length_of_last_word takes a string s as input.

2.Splitting the String:

s.split(): This line splits the input string s into a list of words using whitespace as the delimiter. The split() method without any argument splits the string at whitespace characters (spaces, tabs, and newline characters) and removes leading and trailing whitespaces.

3.Check for Empty List:

if not words:: This line checks if the list of words obtained from the string is empty. If the string s had no words (i.e., it was an empty string or contained only whitespace), the function returns 0 because there is no last word to measure.

4.Return Length of Last Word:

return len(words[-1]): If the list of words is not empty, the function returns the length of the last word in the list. words[-1] accesses the last element in the list, and len(words[-1]) gives the length of that last word.

5.User Input and Printing:

print(length_of_last_word(input())): The code takes user input using the input() function, passes it to the length_of_last_word function, and prints the result.

Medium 2

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

Code:

array.py - C:/Users/034354744/Documents/Deesha/array.py (3.10.11)

File Edit Format Run Options Window Help

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

    candidate1, count1 = None, 0
    candidate2, count2 = None, 0

    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
    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

nums_str = input("Enter the array of integers separated by spaces: ")
nums = list(map(int, nums_str.split()))

print(majority_elements(nums))
```

Output:

```
IDLE Shell 3.10.11
File  Edit  Shell  Debug  Options  Window  Help
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:3
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for m
>>>
============= RESTART: C:/Users/034354744/Documents/Dees
Enter the array of integers separated by spaces: 3 2 3
[3]
>>>
============= RESTART: C:/Users/034354744/Documents/Dees
Enter the array of integers separated by spaces: 1
[1]
>>>
============= RESTART: C:/Users/034354744/Documents/Dees
Enter the array of integers separated by spaces: 1 2
[1, 2]
>>>
```

Logic and algorithm of the code:

1.The algorithm uses the Boyer-Moore Majority Vote algorithm to find potential candidates for majority elements in a single pass through the input list.

2.It then performs a second pass to count the occurrences of these potential candidates.

3.If the count of a candidate is greater than [n/3], where n is the length of the input list, then it is considered a majority element, and it is added to the result list.

4.The final result list contains the majority elements found in the input list.

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*.

Code:

pal.py - C:/Users/034354744/Documents/Deesha/

File Edit Format Run Options Window He

```python
def shortest_palindrome(s):
    n = len(s)
    rev_s = s[::-1]

    for i in range(n):
        if s[:n - i] == rev_s[i:]:
            return rev_s[:i] + s

s = input()

result = shortest_palindrome(s)
print(result)
```

Output:

```
>>>
============== RESTART: C:/Users/0343
aacecaaa
aaacecaaa
>>>
============== RESTART: C:/Users/0343
abcd
dcbabcd
```

Logic and algorithm of the code:

1.The code first calculates the length of the input string s and creates its reverse, rev_s, using slicing (s[::-1]).

2.It then iterates through the original string s using a for loop, starting from index 0.

3.In each iteration, it checks if the substring from the beginning of s to n - i (where i is the loop variable) is a palindrome. It does this by comparing this substring with the reversed substring of the original string, starting from index i in the reversed string (rev_s[i:]).

4.If a palindrome is found, the code returns the concatenation of the reversed substring (rev_s[:i]) and the original string s. This forms the shortest palindrome by adding characters to the beginning of the original string.