Name: Lorraine Becker - ID: 104584773

Assignment 2

Task 1

Test case 1:

Purpose: It will check that it correctly separates them into positive and negative numbers.

Input: [10, -9, 15, 7, -5, 28]
Expected Output:
Positive numbers: [7, 10, 15, 28]
Negative numbers: [-9, -5]

Test case 2:

Purpose: The test case will check whether the function handles the maximum and minimum integer value in the right way, meaning the program can handle full range from minimum to a maximum boundary value and correctly separates and sorts positive and negative integers.

Input: [-100, -50, -1, 1, 50, 100]
Expected Output:
Positive numbers: [1, 50, 100]
Negative numbers: [-100, -50, -1]

Test case 3:

Purpose: This test case is distinct in that it tests a list with only negative numbers, which can verify whether the function only takes negative inputs. It also tests to see if the list of positive integers is empty.

Input: [-4, -2, -7, -1]
Expected Output:
Positive numbers: []
Negative numbers: [-7, -4, -2, -1]

Test case 4:

Purpose: This test case tests a list with only positive numbers. It verifies whether the function works accordingly when inputs contain only positive integers. It also tests to see if the list of negative integers is empty.

Input: [8, 3, 7, 1, 2]
Expected Output:
Positive numbers: [1, 2, 3, 7, 8]
Negative numbers: [ ]

Test case 5:

Purpose: This test case validates the program's handling of zero. Zero should be treated as a negative integer. Therefore, the test ensures that zero is correctly included in the negative list, while also verifying that positive and negative numbers are sorted and handled properly.

Input: [-99, -50, -2, 0, 1, 50, 99]
Expected Output:
Positive numbers: [1, 50, 99]
Negative numbers: [-99, -50, -2, 0]

Test case 6:

Purpose: This test case is comprehensive and covers a larger input list that will handle duplicates and sorting in the program. The elements in the input list are already sorted. It checks to verify if the program keeps this order along with the removal of duplicates.

Input: [-5, -3, -1, 1, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Expected Output:
Positive numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Negative numbers: [-5, -3, -1]

Task 2

If we can only test our program with one test case from Task 1, the most strategic choice would be Test Case 6: [-5, -3, -1, 1, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. Here is the explanation:

1. **Comprehensiveness:** This generic test case covers a wide range of scenarios, including:
    A. Includes both positive and negative integers.
    B. Covers a wide range of values.
    C. Ensures both output lists are non-empty.
    D. Contains duplicates to test handling of non-unique elements.
    E. Includes sorted list, testing the program's sorting functionality.
2. **Complexity:** The test case has significant complexity due to its:
    A. Handling of Duplicates: It tests the program's ability to remove duplicates effectively.
    B. Sorting Functionality: It requires sorting of both positive and negative numbers.
    C. Accurate sorting and separation of positive and negative integers in the output lists.
3. **Risk Deduction:** By choosing a test case that incorporates more conditions and complexities when chosen would lessen the probability of some flaws going undetected. The functionality of the program will be checked rather robustly to ensure sorting and duplicate removal work correctly.

Task 3

Insights of test outcome:

The table below shows that test case 5 and test case 6 failed. Test Case 5 failed because zero was not included in the negative list, and Test Case 6 failed due to duplicate handling issues. These failures indicate problems with duplicate removal and handling of zero.

Input	Output	Expected Output
Test Case 1: [10, -9, 15, 7, -5, 28]	Positive numbers: [7, 10, 15, 28] Negative numbers: [-9, -5]	Positive numbers: [7, 10, 15, 28] Negative numbers: [-9, -5] - Passed
Test Case 2: [-100, -50, -1, 1, 50, 100]	Positive numbers: [1, 50, 100] Negative numbers: [-100, -50, -1]	Positive numbers: [1, 50, 100] Negative numbers: [-100, -50, -1] - Passed
Test Case 3: [-4, -2, -7, -1]	Positive numbers: [] Negative numbers: [-7, -4, -2, -1]	Positive numbers: [] Negative numbers: [-7, -4, -2, -1] - Passed
Test Case 4: [8, 3, 7, 1, 2]	Positive numbers: [1, 2, 3, 7, 8] Negative numbers: [ ]	Positive numbers: [1, 2, 3, 7, 8] Negative numbers: [ ] - Passed

Test Case 5: [-99, -50, -2, 0, 1, 50, 99]	Error: The number 0 is not a valid input.	Positive numbers: [1, 50, 99] Negative numbers: [-99, -50, -2, 0] - Failed
Test Case 6: [-5, -3, -1, 1, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]	Positive numbers: [1, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] Negative numbers: [-5, -3, -1]	Positive numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] Negative numbers: [-5, -3, -1] - Failed

Table 1 - Python program results table

Suggestions to improve the program

1. **Duplicate Handling:** 'set()' was used to eliminate duplicates from the positive and negative lists. While **'**sorted()' ensures that the lists are arranged in ascending order.
2. **Zero as Negative Integer**: Correctly included 0 in the negative list by using 'num <= 0'.
3. **Fixed Syntax Error**: 'set[num for num in nums if num <= 0]' was replaced with 'set(num for num in nums if num <= 0)' to correct the syntax for creating a set. The function now properly handles the input according to the specified requirements and corrects the errors identified in the previous version.
4. **Adjust List Length Check:** Updated the maximum length from 20 to 30 integers.

assignment2.py

```
def split_and_sort(nums):

    # Check input list length
    if len(nums) > 30:
        return "Error: Input list should contain no more than 30 integers."

    # Filter numbers into two separate lists
    pos_nums = sorted(set(num for num in nums if num > 0))

    # Consider 0 as a negative integer
    neg_nums = sorted(set(num for num in nums if num <= 0))

    print("Positive numbers:", pos_nums)
    print("Negative numbers:", neg_nums)

    return neg_nums, pos_nums
```

test.py

```
from assignment2 import split_and_sort

# Test case input
nums = [-5, -3, -1, 1, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

# Call the function and get the results
neg_nums, pos_nums = split_and_sort(nums)

# Print the results
print("Test case 6:")
print("Pos numbers:", pos_nums)
print("Neg numbers:", neg_nums)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/opt/homebrew/bin/python3 /Users/lb008022/Desktop/assignment2-python/test.py
lb008022@Lorraine-Becker---lb008022 assignment2-python % /opt/homebrew/bin/python3 /Users/lb008022/Desktop/assignment2-python/test.py
Positive numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Negative numbers: [-5, -3, -1]
Test case 6:
Pos numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Neg numbers: [-5, -3, -1]
lb008022@Lorraine-Becker---lb008022 assignment2-python %
```

Image 1 & 2 - New Code