Name: Nur E Siam

ID:103842784

Assignment 2

Task 1:

Here are six valid input test cases, each serving a distinct testing purpose, along with their justifications. This test cases are below:

Test Case ID	Test Case	Output	Purpose	Justification
1	[-100, -99, -98, -97]	Positive list: [], Negative list: [-100, -99, -98, -97]	To test the behavior of the program when given the smallest allowable integer values.	This ensures the program handles boundary values at the negative extreme and correctly returns the sorted output
2	[34, 0, -1, 23, -45, 0, 89, 1]	Positive list: [1, 23, 34, 89], Negative list: [-45, -1, 0]	To test how the program handles zero values (treated as negative) and mixed numbers with both positive and negative values.	This case checks the program's ability to treat zero as a negative number and handle multiple zeros appropriately.
3	[50, -50, 50, -50, 50, -50]	Positive list: [50], Negative list: [-50]	To test the program's handling of repeated positive and negative numbers.	This case examines if the program removes duplicates from both positive and negative lists while maintaining order.
4	[10, 20, 30, 40, 50]	Positive list: [10, 20, 30, 40, 50], Negative	To test the scenario where the input list	This checks if the program correctly

		list: []	contains only positive numbers.	handles an input without negative numbers and returns an empty list for negatives.
5	[10, -9, 15, 7, 10, -5, 28]	Positive list: [7, 10, 15, 28], Negative list: [-9, -5]	This case includes both positive and negative integers with duplicate values	This tests that the program correctly handles mixed integers with duplicates, ensuring it filters and sorts both lists properly.
6	[i for i in range(-15, 15)]	Positive list: [1, 2, ..., 14], Negative list: [-15, -14, ..., -1, 0]	Tests the program with the maximum number of integers allowed in the input list.	This tests the upper limit of input list size and ensures the program does not exceed the limit of 30 integers.

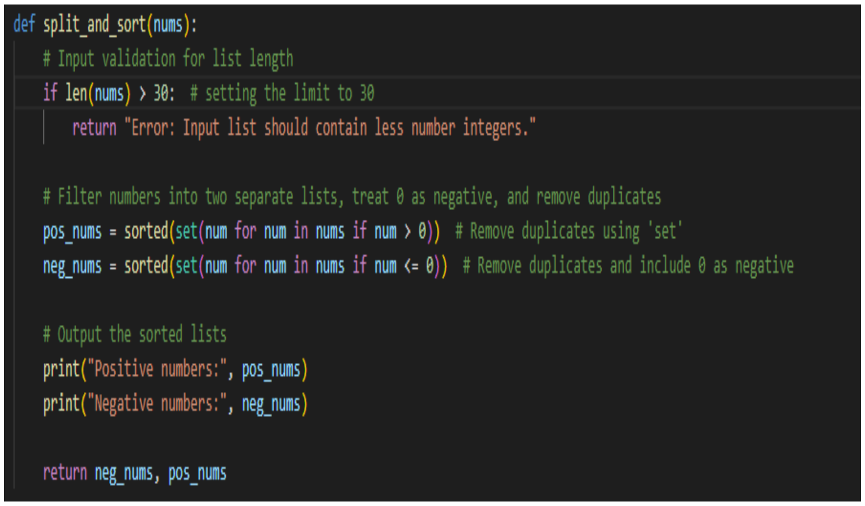
Task 2: Selection of a Single Test Case

If I had to choose just one test case to run due to resource constraints, I would select Test Case 2 because it covers multiple aspects. The selected test case covers a diverse range of input values, including positive numbers, negative numbers, and zeros, allowing for testing multiple features at once. Zero is treated as a negative number per the program's logic, ensuring proper handling of edge cases. It also includes duplicate numbers, letting us verify whether the program correctly filters them and ensures each value appears only once.

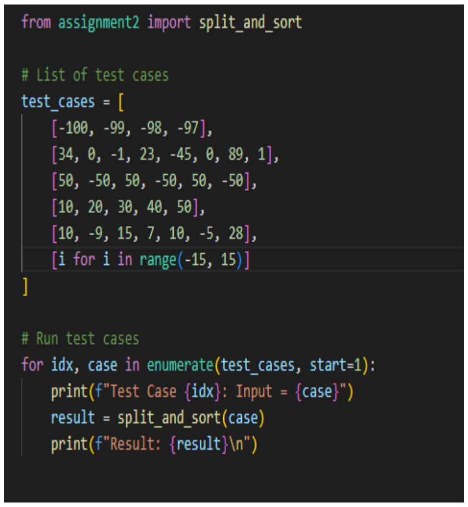
Additionally, the use of metamorphic relations strengthens the test by modifying input values to check if the program produces consistent outputs. This approach tests the program's reliability in handling input transformations while ensuring sorting accuracy, removal of duplicates, and correct treatment of zeros, making it a comprehensive test for uncovering any fundamental issues in the program's logic.

Task 3:

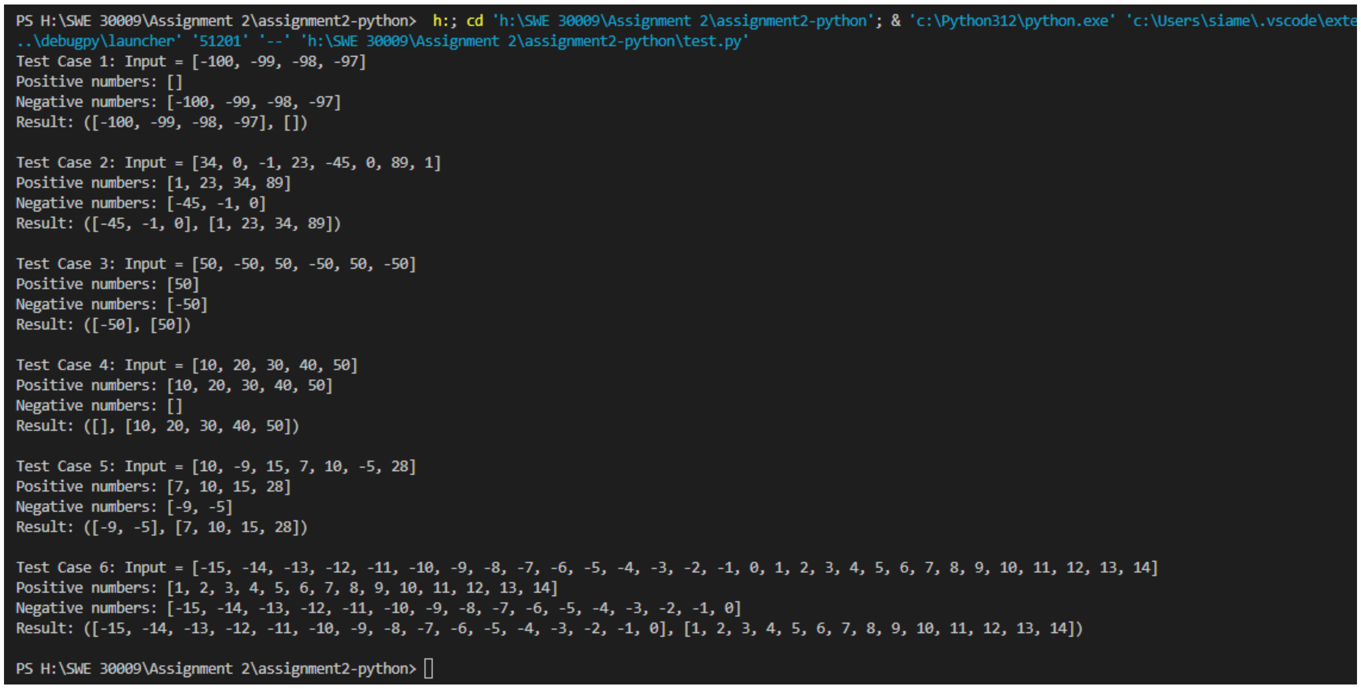
After implementing the necessary fixes (removing duplicates, allowing up to 30 integers, and treating 0 as a negative number), the following Python script was used to run the provided test cases.



```python
def split_and_sort(nums):
    # Input validation for list length
    if len(nums) > 30:  # setting the limit to 30
        return "Error: Input list should contain less number integers."

    # Filter numbers into two separate lists, treat 0 as negative, and remove duplicates
    pos_nums = sorted(set(num for num in nums if num > 0))  # Remove duplicates using 'set'
    neg_nums = sorted(set(num for num in nums if num <= 0))  # Remove duplicates and include 0 as negative

    # Output the sorted lists
    print("Positive numbers:", pos_nums)
    print("Negative numbers:", neg_nums)

    return neg_nums, pos_nums
```



```python
from assignment2 import split_and_sort

# List of test cases
test_cases = [
    [-100, -99, -98, -97],
    [34, 0, -1, 23, -45, 0, 89, 1],
    [50, -50, 50, -50, 50, -50],
    [10, 20, 30, 40, 50],
    [10, -9, 15, 7, 10, -5, 28],
    [i for i in range(-15, 15)]
]

# Run test cases
for idx, case in enumerate(test_cases, start=1):
    print(f"Test Case {idx}: Input = {case}")
    result = split_and_sort(case)
    print(f"Result: {result}\n")
```



```
PS H:\SWE 30009\Assignment 2\assignment2-python>  h:; cd 'h:\SWE 30009\Assignment 2\assignment2-python'; & 'c:\Python312\python.exe' 'c:\Users\siame\.vscode\exte
..\debugpy\launcher' '51201' '--' 'h:\SWE 30009\Assignment 2\assignment2-python\test.py'
Test Case 1: Input = [-100, -99, -98, -97]
Positive numbers: []
Negative numbers: [-100, -99, -98, -97]
Result: ([-100, -99, -98, -97], [])

Test Case 2: Input = [34, 0, -1, 23, -45, 0, 89, 1]
Positive numbers: [1, 23, 34, 89]
Negative numbers: [-45, -1, 0]
Result: ([-45, -1, 0], [1, 23, 34, 89])

Test Case 3: Input = [50, -50, 50, -50, 50, -50]
Positive numbers: [50]
Negative numbers: [-50]
Result: ([-50], [50])

Test Case 4: Input = [10, 20, 30, 40, 50]
Positive numbers: [10, 20, 30, 40, 50]
Negative numbers: []
Result: ([], [10, 20, 30, 40, 50])

Test Case 5: Input = [10, -9, 15, 7, 10, -5, 28]
Positive numbers: [7, 10, 15, 28]
Negative numbers: [-9, -5]
Result: ([-9, -5], [7, 10, 15, 28])

Test Case 6: Input = [-15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Positive numbers: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Negative numbers: [-15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0]
Result: ([-15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0], [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])

PS H:\SWE 30009\Assignment 2\assignment2-python> 
```

The program successfully handled all test cases after fixing the issues with duplicates, input size, and zero handling. It correctly removed duplicates, sorted inputs, and treated zero as negative, demonstrating its ability to manage various input scenarios effectively.