Assignment 2

SWE30009 Software Testing and Reliability

Student Name: Nguyen Nam Tung || **Student ID:** 103181157

Task 1:

No	Input	Expected Output	Purpose	Justification
1	[3, 4, 2, 1]	Positive numbers: [1, 2, 3, 4] Negative numbers: []	The purpose of this test case is to check if the program correctly handles and sorts a list containing only positive integers in ascending order and identifies that there are no negative integers.	The input list contains only positive integers, without duplication, in unsorted order. The program should correctly produce a list of only positive numbers in ascending order, while an empty list for negative numbers.
2	[-3, -4, -1, -2]	Positive numbers: [] Negative numbers: [-4, -3, -2, -1]	This test case ensures that the program correctly handles a list of only negative integers, sorts them in ascending order, and correctly identifies that there are no positive integers	The input list contains only negative integers, without duplication, in unsorted order. The program should correctly produce a list of only negative numbers in ascending order, while an empty list for negative numbers.
3	[1, 1, 2, 4]	Positive numbers: [1,2,4] Negative numbers: []	The purpose of this test case is to check the program's ability to handle a list of duplicate positive integers, ensuring those duplicates are removed.	The input list contains only positive numbers in ascending order but with duplication. The program should correctly produce a list containing only unique positive integers without duplication and an empty list for the negative numbers.
4	[-4, -4, -2, -1]	Positive numbers: [] Negative numbers: [-4, -2, -1]	The purpose of this test case is to check the program’s ability to handle a list of duplicate negative integers, ensuring those duplicates are removed	The input list contains only negative numbers in ascending order but with duplication. The program should correctly produce a list containing only unique negative integers without duplication and an empty list for the positive numbers.
5	[-2, -5, -4, 3, 4, 3, -2, 5]	Positive numbers: [3, 4, 5] Negative numbers:	The purpose of this test case is to validate the program's ability to handle a mixed list of	The input list contains both negative and positive integers, with duplication and they are all

		[-5, -4, -2]	positive and negative integers, including duplicates of both, ensuring correct sorting and deduplication	in unsorted order. The program should correctly produce 2 lists: one for positive numbers and one for negative numbers. They all contain no duplication, and the numbers are in ascending order.
6	[0]	Positive numbers: [] Negative numbers: [0]	The purpose of this test case is to check if the program can consider the number 0 as a negative number.	The input list contains only the number 0, which should be treated as a normal negative. The program should correctly produce a list of negative numbers that contain only the number 0, and an empty list for the positive numbers.

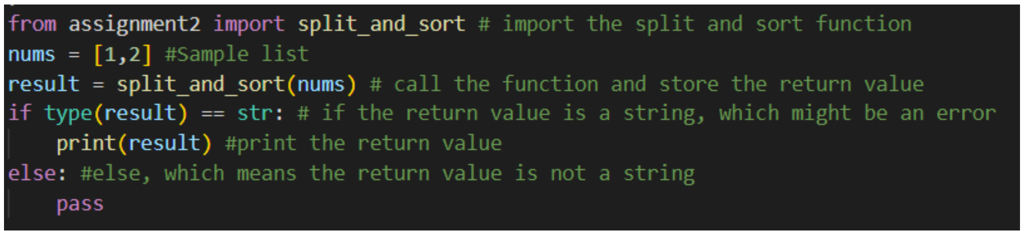
Table 1: Concrete test cases

Task 2:

The test case 5, which has the input of [-2, -5, -4, 3, 4, 3, -2, 5] and an expected output: Positive numbers: [3, 4, 5] Negative numbers: [-5, -4, -2] is the most optimal test case for several reasons. First of all, the test case comprises a mix of both positive and negative numbers, which can test the program's ability to separate the input into two lists of positive and negative numbers. Additionally, the test case comprises of duplicated numbers, which can be used to test the program's functionality to remove the duplicate values. Lastly, the test case contains values that are not in the correct ascending order, which can test the program's capability to sort the numbers in the correct order. Overall, test case 5 should be utilized as it can cover the testing of the basic functionalities of the program.

Task 3:

Below is the Python test script, which was utilized for the testing of all the test cases that have been identified in the aforementioned section. I have made some small modifications to the default test script provided, to ensure that when a return value is a string (which might be a message that displays the error), it can be printed out to the console.



```python
from assignment2 import split_and_sort # import the split and sort function
nums = [1,2] #Sample list
result = split_and_sort(nums) # call the function and store the return value
if type(result) == str: # if the return value is a string, which might be an error
    print(result) #print the return value
else: #else, which means the return value is not a string
    pass
```

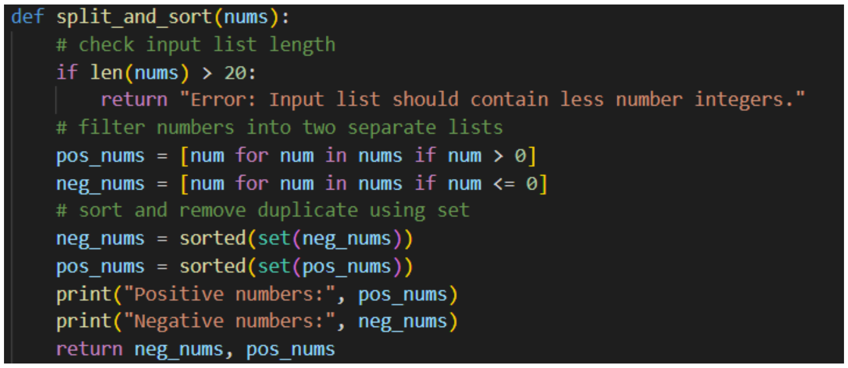
Figure 1: Test Script Used in the assignment

No	Input	Expected Output	Actual Output	Result
1	[3, 4, 2, 1]	Positive numbers: [1, 2, 3, 4] Negative numbers: []	Positive numbers: [1, 2, 3, 4] Negative numbers: []	Passed
2	[-3, -4, -1, -2]	Positive numbers: [] Negative numbers: [-4, -3, -2, -1]	Positive numbers: [] Negative numbers: [-4, -3, -2, -1]	Passed
3	[1, 1, 2, 4]	Positive numbers: [1,2,4] Negative numbers: []	Positive numbers: [1, 1, 2, 4] Negative numbers: []	Failed
4	[-4, -4, -2, -1]	Positive numbers: [] Negative numbers: [-4, -2, -1]	Positive numbers: [] Negative numbers: [-4, -4, -2, -1]	Failed
5	[-2, -5, -4, 3, 4, 3, -2, 5]	Positive numbers: [3, 4, 5] Negative numbers: [-5, -4, -2]	Positive numbers: [3, 3, 4, 5] Negative numbers: [-5, -4, -2, -2]	Failed
6	[0]	Positive numbers: [] Negative numbers: [0]	Error: The number 0 is not a valid input.	Failed

Table 2: Test results

From the test outputs, it can be seen that the program can successfully sort the output in ascending order and produce two separate lists for both positive and negative numbers. If the input list only contains negative or positive numbers, an empty list is also correctly produced for the type with no numbers.

However, the program failed to remove duplicates for both the negative and positive integers, and it also failed to recognize the number 0 as a negative number. As a program, the program should be modified to check and treat the number 0 as a negative number. Meanwhile, the duplication removal functionality can also be applied using a “set” function in Python. This function converts the input list into a set, which can automatically remove all the duplicate elements in a list. Below is the modified program to pass the tests:



```python
def split_and_sort(nums):
    # check input list length
    if len(nums) > 20:
        return "Error: Input list should contain less number integers."
    # filter numbers into two separate lists
    pos_nums = [num for num in nums if num > 0]
    neg_nums = [num for num in nums if num <= 0]
    # sort and remove duplicate using set
    neg_nums = sorted(set(neg_nums))
    pos_nums = sorted(set(pos_nums))
    print("Positive numbers:", pos_nums)
    print("Negative numbers:", neg_nums)
    return neg_nums, pos_nums
```

Figure 2: The improved and modified code for the program to pass the test

After implementing the modifications, all tests were re-run, and every test case passed successfully. This confirms that the program now fully meets all the intended objectives and handles all the scenarios outlined in the aforementioned test cases effectively.