SWE30009 – Semester May, 2024.

Name: Phan Vu – Student ID: 104222099

Assignment 2: Software Testing and Reliability

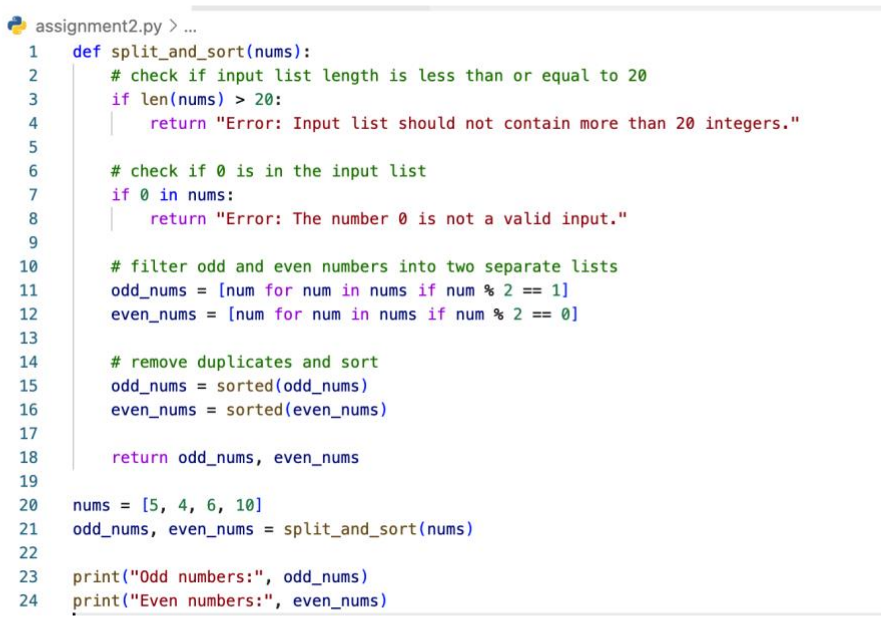
Swinburne University of Technology

Introduction

This assignment aims to strengthen your understanding of software testing activities and the process of generating test cases for a given program. The focus is on detecting any possible issues with the provided split_and_sort function, which splits an input list of integers into two lists of odd and even integers.

Program Under Test

assignment2.py



assignment2.py > ...

```python
def split_and_sort(nums):
    # check if input list length is less than or equal to 20
    if len(nums) > 20:
        return "Error: Input list should not contain more than 20 integers."

    # check if 0 is in the input list
    if 0 in nums:
        return "Error: The number 0 is not a valid input."

    # filter odd and even numbers into two separate lists
    odd_nums = [num for num in nums if num % 2 == 1]
    even_nums = [num for num in nums if num % 2 == 0]

    # remove duplicates and sort
    odd_nums = sorted(odd_nums)
    even_nums = sorted(even_nums)

    return odd_nums, even_nums

nums = [5, 4, 6, 10]
odd_nums, even_nums = split_and_sort(nums)

print("Odd numbers:", odd_nums)
print("Even numbers:", even_nums)
```

Objective

The main testing objective is to ensure the function correctly splits the input list into two lists (one containing odd integers and the other containing even integers), removes duplicates, and sorts the integers in ascending order.

Task 1: Construct Six Valid Concrete Test Cases

Test Cases

Test Case Number	*Input List*	*Expected Output (Odd)*	*Expected Output (Even)*	*Purpose*
1	[3, 10, 9, 20, 16, 10, 9, 15, 7, 5, 28, 20, 5, 8, 20]	[3, 5, 7, 9, 15]	[8, 10, 16, 20, 28]	To test the program's ability to handle typical inputs with duplicates.
2	[13, 7, 9, 15, 21, 5, 3]	[3, 5, 7, 9, 13, 15, 21]	[]	To test the scenario where there are no even numbers in the input.
3	[4, 8, 12, 16, 20, 2, 10]	[]	[2, 4, 8, 10, 12, 16, 20]	To test the scenario where there are no odd numbers in the input.

4	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]	[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]	[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]	To test the upper boundary condition with 20 elements.
5	[5, 5, 5, 10, 10, 15, 15, 20, 20, 25, 25]	[5, 15, 25]	[10, 20]	To test how the program handles multiple duplicates.
6	[19, 22, 5, 8, 12, 25, 30]	[5, 19, 25]	[8, 12, 22, 30]	To test with a completely random set of positive numbers.

Task 2: Choose and Justify a Single Test Case

Chosen Test Case

- **Test Case Number:** 4
- **Input List:** [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
- **Expected Output:** ([1, 3, 5, 7, 9, 11, 13, 15, 17, 19], [2, 4, 6, 8, 10, 12, 14, 16, 18, 20])

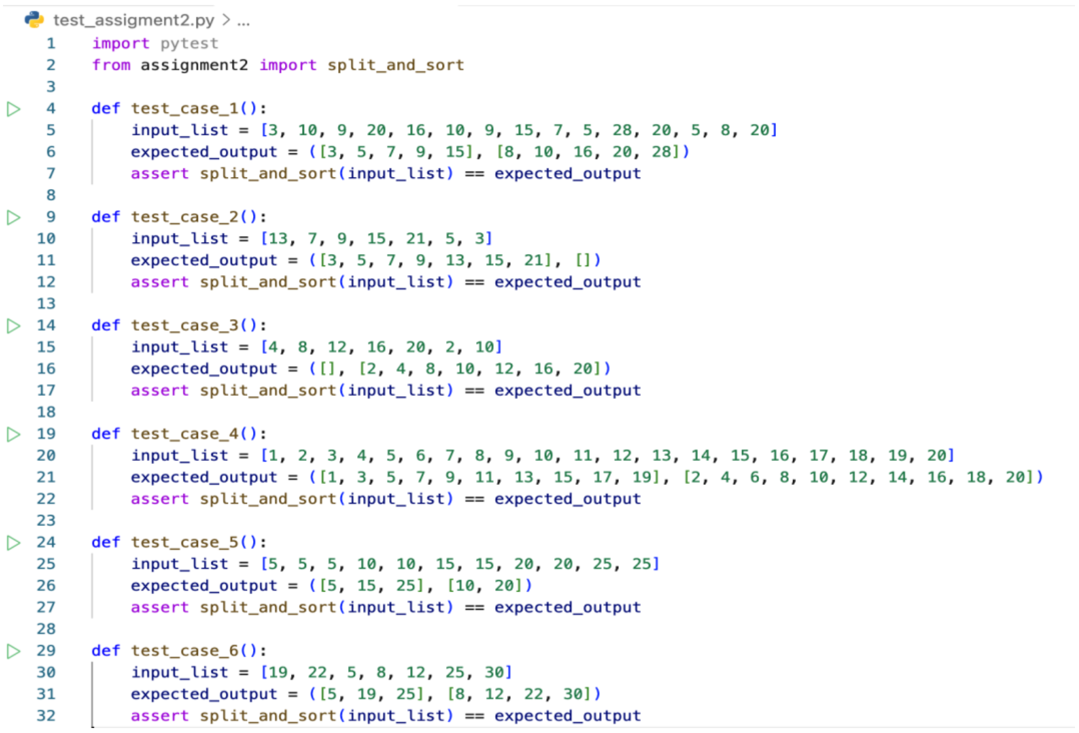
Justification

This test case comprehensively covers the boundary condition with the maximum number of allowed elements. It includes both odd and even numbers, ensuring the function's capability to separate and sort them correctly.

Task 3: Conduct the Testing

Testing Script

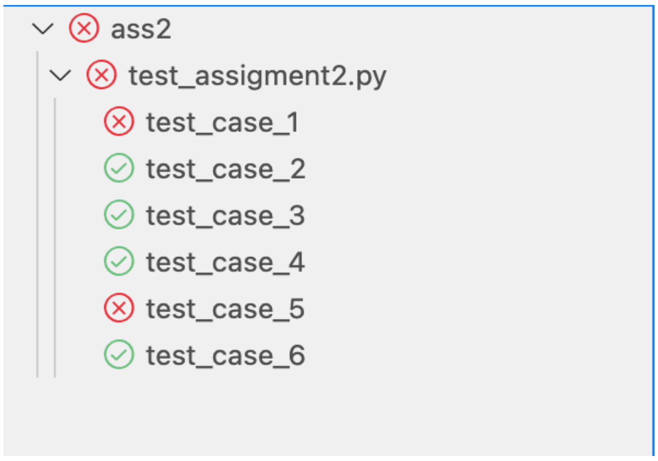
Here is the testing script using ***pytest*** to validate the ***split_and_sort*** function:



```python
import pytest
from assignment2 import split_and_sort

def test_case_1():
    input_list = [3, 10, 9, 20, 16, 10, 9, 15, 7, 5, 28, 20, 5, 8, 20]
    expected_output = ([3, 5, 7, 9, 15], [8, 10, 16, 20, 28])
    assert split_and_sort(input_list) == expected_output

def test_case_2():
    input_list = [13, 7, 9, 15, 21, 5, 3]
    expected_output = ([3, 5, 7, 9, 13, 15, 21], [])
    assert split_and_sort(input_list) == expected_output

def test_case_3():
    input_list = [4, 8, 12, 16, 20, 2, 10]
    expected_output = ([], [2, 4, 8, 10, 12, 16, 20])
    assert split_and_sort(input_list) == expected_output

def test_case_4():
    input_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
    expected_output = ([1, 3, 5, 7, 9, 11, 13, 15, 17, 19], [2, 4, 6, 8, 10, 12, 14, 16, 18, 20])
    assert split_and_sort(input_list) == expected_output

def test_case_5():
    input_list = [5, 5, 5, 10, 10, 15, 15, 20, 20, 25, 25]
    expected_output = ([5, 15, 25], [10, 20])
    assert split_and_sort(input_list) == expected_output

def test_case_6():
    input_list = [19, 22, 5, 8, 12, 25, 30]
    expected_output = ([5, 19, 25], [8, 12, 22, 30])
    assert split_and_sort(input_list) == expected_output
```

Test Results

After running the test, here are the results:



```
ass2 ✗
  test_assigment2.py ✗
    ✗ test_case_1
    ✓ test_case_2
    ✓ test_case_3
    ✓ test_case_4
    ✗ test_case_5
    ✓ test_case_6
```

Insights and Program Improvement

Insights:

- The test results indicate that four out of the six test cases passed successfully, demonstrating that the split_and_sort function can handle typical inputs correctly, inputs with only odd or only even numbers, and a random set of positive numbers.
- However, two test cases failed. Specifically:
  - **Test Case 1:** The function did not remove duplicates correctly before sorting, as shown by the presence of repeated elements in the output.
  - **Test Case 5:** Similar to Test Case 1, the function failed to remove duplicates, resulting in repeated elements in the output.

Program Improvement:

- **Duplicate Removal:** The primary issue causing the test failures is the function's inability to remove duplicates before sorting the integers. This can be resolved by using a set to filter out duplicates before sorting. Here's a suggestion for improvement:

Converting the list to a set before sorting ensures all duplicates are removed, addressing the failures observed in Test Cases 1 and 5.

- **Boundary Checking:** While the current function checks for the length of the input list and the presence of zero, additional edge cases should be considered, such as negative integers and large input values within the 20-element limit, to ensure comprehensive testing and robustness.