

Assignment 2 Report

Student Name: Lau Ngoc Quyen

Student ID: 104198996

Task 1: Development of Fifth Test Cases For Sorting and Filtering Numbers

Test Case 1: Only Even Numbers

Source Input (SI): [2, 4, 6, 8, 16, 18, 20]

Testing Purposes: This test case consists entirely of even integers. It tests the scenario where the output list of odd integers should be empty.

Source Output (SO): - Odd numbers: [] - Even numbers: [2, 4, 6, 8, 16, 18, 20]

Metamorphic Relation (MR): Addition of a Constant to an Element

Modification: Add a constant value k to the first element of SI ($k = 22$)

Follow-up Input (FI): [2, 4, 6, 8, 16, 18, 20, 22]

Follow-up Output (FO): - Odd numbers: [] - Even numbers: [2, 4, 6, 8, 16, 18, 20, 22]

Test Case 2: Only Odd Numbers

Source Input (SI): [1, 3, 5, 7, 9, 11, 15, 17]

Testing Purposes: This test case consists entirely of odd integers. It tests the scenario where the output list of even integers should be empty.

Source Output (SO): - Odd numbers: [1, 3, 5, 7, 9, 11, 15, 17] - Even numbers: []

Metamorphic Relation (MR): Permutation

Modification: Permute the elements of SI

Follow-up Input (FI): [3, 5, 7, 11, 17, 15, 1, 9]

Follow-up Output (FO): - Odd numbers: [1, 3, 5, 7, 9, 11, 15, 17] - Even numbers: []

Test Case 3: Combination Of Odd and Even Numbers With Duplicates

Source Input (SI): [1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13]

Testing Purposes: The test case involves dividing an output list of odd and even numbers, removing duplicates in ascending order, into odd and even numbers.

Source Output (SO): - Odd numbers: [1, 3, 5, 7, 9, 11, 13] - Even numbers: [2, 4, 6, 8, 10, 12]

Metamorphic Relation (MR): Duplication

Modification: Duplicate SI

Follow-up Input (FI): [2, 2, 2, 3, 4, 4, 5, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13, 13]

Follow-up Output (FO): - Odd numbers: [1, 3, 5, 7, 9, 11, 13] - Even numbers: [2, 4, 6, 8, 10, 12]

Test Case 4: Large List With Mixed Number

Source Input (SI): [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Testing Purposes: This test case contains a large number of mixed odd and even integers with duplicates. It tests the performance and correctness of the algorithm on larger inputs.

Source Output (SO): - Odd numbers: [17, 19, 21, 23, 25, 27, 29] - Even numbers: [18, 20, 22, 24, 26, 28, 30]

Metamorphic Relation (MR): Concatenation

Modification: Concatenate SI with itself

Follow-up Input (FI): [15, 15, 16, 16, 17, 17, 18, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Follow-up Output (FO): - Odd numbers: [15, 17, 19, 21, 23, 25, 27, 29] - Even numbers: [16, 18, 20, 22, 24, 26, 28, 30]

Test Case 5: Only One Odd Number

Source Input (SI): [3]

Testing Purposes: This test case contains an single odd number

Source Output (SO): - Odd numbers: [3] - Even numbers: []

Metamorphic Relation (MR): Addition of a Constant to an Element

Modification: Add a constant value k to the first element of SI ($k = 2$)

Follow-up Input (FI): [3, 2]

Follow-up Output (FO): - Odd numbers: [3] - Even numbers: [2]

Test Case 6: Only One Even Number

Source Input (SI): [2]

Testing Purposes: This test case contains an single even number

Source Output (SO): - Odd numbers: [] - Even numbers: [2]

Metamorphic Relation (MR): Addition of a Constant to an Element

Modification: Add a constant value k to the first element of SI (k = 1)

Follow-up Input (FI): [1, 2]

Follow-up Output (FO): - Odd numbers: [1] - Even numbers: [2]

Task 2: Selection Of Single Test Case

Selected Test Case:

Test Case 3: Combination Of Odd and Even Numbers With Duplicates

Test Case: [1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13]

Expected Output: - Odd numbers: [1, 3, 5, 7, 9, 11, 13] - Even numbers: [2, 4, 6, 8, 10, 12]

Justification:

- **Diverse Input:** Contains both odd and even numbers.
- **Duplicate Handling:** Includes duplicates (e.g., 1, 2, 3, 4, 5) to test removal.
- **Sorting Check:** Verifies sorting of both odd and even numbers.
- **Comprehensive Coverage:** Tests key functionalities—separation, duplication removal, and sorting.
- **Input Limitation:** 17 elements, well within the 20-element limit, ensuring adequate performance testing.

=> This test case effectively evaluates the core features of the program with limited testing resources.

Task 3: Testing, Analysis And Suggestion For Improving of the Program

Test Case 1: Only Even Numbers

Test Case: [2, 4, 6, 8, 16, 18, 20]

Actual Result: - Odd numbers: [] - Even numbers: [2, 4, 6, 8, 16, 18, 20]

Test Case 2: Only Odd Numbers

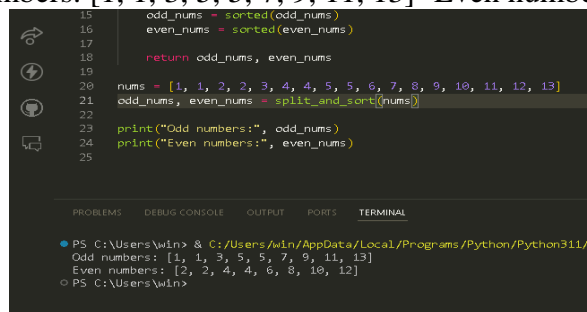
Test Case: [1, 3, 5, 7, 9, 11, 15, 17]

Actual Result: - Odd numbers: [1, 3, 5, 7, 9, 11, 15, 17] - Even numbers: []

Test Case 3: Combination Of Odd and Even Numbers With Duplicates

Test Case: [1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13] (**Error Program**)

Actual Result: - Odd numbers: [1, 1, 3, 5, 5, 7, 9, 11, 13] -Even numbers: [2, 2, 4, 4, 6, 8, 10, 12]



```
15 odd_nums = sorted(odd_nums)
16 even_nums = sorted(even_nums)
17
18 return odd_nums, even_nums
19
20 nums = [1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12, 13]
21 odd_nums, even_nums = split_and_sort(nums)
22
23 print("Odd numbers:", odd_nums)
24 print("Even numbers:", even_nums)
25
```

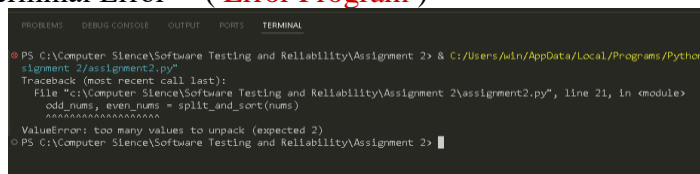
PROBLEMS DEBUG CONSOLE OUTPUT PORTS TERMINAL

```
PS C:\Users\win> & C:\Users\win\AppData\Local\Programs\Python\Python311\p
Odd numbers: [1, 1, 3, 5, 5, 7, 9, 11, 13]
Even numbers: [2, 2, 4, 4, 6, 8, 10, 12]
PS C:\Users\win>
```

Test Case 4: Large List With Mixed Number

Test Case: [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

Actual Result: - "Terminal Error " (**Error Program**)



```
PROBLEMS DEBUG CONSOLE OUTPUT PORTS TERMINAL
PS C:\Computer Science\Software Testing and Reliability\Assignment 2> & C:\Users\win\AppData\Local\Programs\Python\
signment 2\assignment2.py
Traceback (most recent call last):
  File "C:\Computer Science\Software Testing and Reliability\Assignment 2\assignment2.py", line 21, in <module>
    odd_nums, even_nums = split_and_sort(nums)
    ^^^^^^^^^^^^^^^^^^^^^
ValueError: too many values to unpack (expected 2)
PS C:\Computer Science\Software Testing and Reliability\Assignment 2>
```

Test Case 5: Only One Odd Number

Test Case: [3]

Actual Result: Odd numbers: [3]

Test Case 6: Only One Even Number

Test Case: [2]

Actual Result: Even numbers: [2]

Discussions And Insights:

- **Correctness:** Although the function `split_and_sort(nums)` correctly handles valid input cases by splitting the list into odd and even numbers, it sorts the lists in ascending order. It does not remove all the odd and even duplicates.
- **Error Handling:** The function unsuccessfully detects and handles errors for inputs lists with more than 20 elements which lead to the terminal error in the text editor

Suggestion For Program Improvement:

1. Using `set()`

```
13
14     # remove duplicates and sort
15     odd_nums = sorted(set(odd_nums))
16     even_nums = sorted(set(even_nums))
17
```

- The `set()` function converts lists of numbers into sets, removing duplicates, ensuring unique values in `odd_nums` and `even_nums` lists. The `sorted()` function sorts elements in ascending order, meeting problem statement requirements.

2. Using the `if()` and `else()` statements

```
if isinstance(result, str):
    print(result)
else:
    odd_nums, even_nums = result
    print("Odd numbers:", odd_nums)
    print("Even numbers:", even_nums)
```

- The `split_and_sort()` function returns either a string or a tuple containing `odd_nums` and `even_nums` lists, depending on whether the function was successful or not. The if-else statement handles both cases, ensuring the user receives the appropriate output.

```
# check if input list is empty
if len(nums) == 0:
    return "Error: Please enter numbers."
```

- `len(nums)` evaluates whether the length of the input list `nums` is zero, meaning the list has no elements and it will return an error for an empty input .

3. Loop through the indices of the list using `for()` condition

```
# check if input list contains negative numbers
for i in range(len(nums)):
    if nums[i] < 0:
        return "Error: The number should not be negative."
```

- This check iterates over each element in the input list `nums` to see if there are any negative numbers loops through the indices of the list. This check iterates over each element in the input list `nums` to see if there are any negative numbers.

Link to code: <https://github.com/quynezz/SWE30009>