SWE30009 – Software Testing and Reliability

Assignment 2

Student name: Vi Luan Dang

Student ID: 103802759

Task 1:

Testcases for task 1 are as follows:

Input	Output A	Output B	Justification
Test case 1: [5, 14, 3, 20, 9, 14, 5, 7, 18, 16, 10, 20, 9, 8, 3]	**Odd integers list**: [3, 5, 7, 9]	**Even integers list**: [8, 10, 14, 16, 18, 20]	Includes both odd and even integers with duplicates to ensure that the program correctly removes duplicates.
Test case 2: [2, 4, 6, 8, 2, 4, 6, 8, 10, 12]	**Odd integers list**: []	**Even integers list**: [2, 4, 6, 8, 10, 12]	Includes only even integers, with duplicates, to ensure that the program can handles a list with no odd integers, removes duplicates.
Test case 3: [13, 7, 9, 15, 3, 3, 7, 9]	**Odd integers list**: [3, 7, 9, 13, 15]	**Even integers list**: []	Includes only odd integers, with duplicates, to ensure that the program can handles a list with no even integers, removes duplicates.
Test case 4: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]	**Odd integers list**: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]	**Even integers list**: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]	Includes the maximum number of elements to ensure that the program can handle the upper boundary condition.
Test case 5: [5]	**Odd integers list**: [5]	**Even integers list**: []	Includes the minimum number of elements to ensure that the program can handle the lower boundary condition.
Test case 6: [999999999, 888888888, 777777777, 666666666]	**Odd integers list**: [777777777, 999999999]	**Even integers list**: [666666666, 888888888]	Includes four large integer values to check if the program can handle and correctly process large numbers.

Task 2:

Given the constraints of only being able to test the program with only one of the above test cases, the most comprehensive and representative test case would be:

Chosen Test Case: Mixed Odd and Even with Duplicates.

Example Input: [5, 14, 3, 20, 9, 14, 5, 7, 18, 16, 10, 20, 9, 8, 3]

Outputs:

- **Odd integers list**: [3, 5, 7, 9]
- **Even integers list**: [8, 10, 14, 16, 18, 20]

Justification:

- **Mix of Odd and Even Integers**: This test case includes both odd and even integers, which ensures that the program correctly identifies and separates odd and even numbers.
- **Duplicates**: The presence of duplicate values tests whether the program can properly remove duplicates.
- **Sorting**: With a variety of numbers, this test case checks if the program correctly sorts the resulting lists in ascending order.
- **Complexity**: This test case is more complex than simpler cases (like only odd or only even numbers) and thus covers more potential edge cases.

Explanation of Choice:

This test case provides a comprehensive test of the program's functionality by combining multiple aspects into a single test:

- It tests the ability to handle mixed data types (odd and even numbers).
- It ensures duplicates are managed correctly.
- It checks the sorting mechanism for both lists.
- It validates the program's behavior with a moderate amount of data (less than the maximum of 20 elements), ensuring typical use case handling.

Task 3:

After conducting testing using Task 1's test cases on the program, the result are as follows:

Test case	Expected Output	Actual Output	Status
Test case 1: [5, 14, 3, 20, 9, 14, 5, 7, 18, 16, 10, 20, 9, 8, 3]	**Odd integers list**: [3, 5, 7, 9] **Even integers list**: [8, 10, 14, 16, 18, 20]	**Odd integers list**: [3, **3**, 5, **5**, 7, 9, 9] **Even integers list**: [8, 10, 14, **14**, 16, 18, 20, **20**]	**Failed to remove duplicates**
Test case 2: [2, 4, 6, 8, 2, 4, 6, 8, 10, 12]	**Odd integers list**: [] **Even integers list**: [2, 4, 6, 8, 10, 12]	**Odd integers list**: [] **Even integers list**: [2, **2**, 4, **4**, 6, **6**, 8, **8**, 10, 12]	**Failed to remove duplicates**

Test case 3: [13, 7, 9, 15, 3, 3, 7, 9]	**Odd integers list**: [3, 7, 9, 13, 15] **Even integers list**: []	**Odd integers list**: [3, 3, 7, 7, 9, 9, 13, 15] **Even integers list**: []	**Failed to remove duplicates**
Test case 4: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]	**Odd integers list**: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19] **Even integers list**: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]	**Odd integers list**: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19] **Even integers list**: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]	**Passed**
Test case 5: [5]	**Odd integers list**: [5] **Even integers list**: []	**Odd integers list**: [5] **Even integers list**: []	**Passed**
Test case 6: [999999999, 888888888, 777777777, 666666666]	**Odd integers list**: [777777777, 999999999] **Even integers list**: [666666666, 888888888]	**Odd integers list**: [777777777, 999999999] **Even integers list**: [666666666, 888888888]	**Passed**

Based on the results from the conducted test cases, it is evident that the current implementation of the program fails to remove duplicate values as expected. Upon reviewing the source code, it becomes apparent that the program lacks functionality for handling duplicates effectively. To address this issue, it is recommended to utilize Python's `*Set*` data structure, which inherently stores only unique elements, ensuring that duplicate integers are properly eliminated before sorting the resulting lists.

Figure 1: Modified program for removing duplicate values.