**Test Plan: Numbers to Words Converter Program**

Introduction:

This Number to Word Converter program converts numerical input into words.  
The purpose of this test case is to ensure the accuracy, functionality, and robustness of the converter program.

Scope:

This test plan covers testing for all aspects of the Numbers to Words Converter program, including input validation, conversion accuracy, edge cases, and error handling.

Test Objectives:

* Verify that the converter accurately converts the numerical input into the desired words.
* Validate the program's handling of different input types (e.g., integers, decimals).
* Asses the performance and scalability of the converter for large numbers.

Test Environment:

* Programming Language: C#
* Development Environment: Visual studio
* Testing Framework: MSTest
* Operating System: Windows

Test Cases:

*Test Case 1: Convert Small Number Integer*

* Input: 1
* Expected Output: "ONE DOLLARS"

*Test Case 2: Convert Big Number Integer*

* Input: 999999999999999999
* Expected Output: "NINE HUNDRED AND NINETY-NINE QUADRILLION NINE HUNDRED AND NINETY-NINE TRILLION NINE HUNDRED AND NINETY-NINE BILLION NINE HUNDRED AND NINETY-NINE MILLION NINE HUNDRED AND NINETY-NINE THOUSAND NINE HUNDRED AND NINETY-NINE DOLLARS"

*Test Case 3: Convert Small Decimal Number Integer*

* Input: 00.01
* Expected Output: "ONE CENTS"

*Test Case 4: Convert Big Decimal Number Integer*

* Input: 00.99
* Expected Output: "NINETY-NINE CENTS"

*Test Case 5: Convert Decimal Number*

* Input: 1234.56
* Expected Output: "ONE THOUSAND TWO HUNDRED AND THIRTY-FOUR DOLLARS AND FIFTY-SIX CENTS"

*Test Case 6: Convert Thousand Number*

* Input: 1000.00
* Expected Output: "ONE THOUSAND DOLLARS"

*Test Case 7: Convert Million Number*

* Input: 1000000.00
* Expected Output: "ONE MILLION DOLLARS"

*Test Case 8: Convert Billion Number*

* Input: 1000000000.00
* Expected Output: "ONE BILLION DOLLARS"

*Test Case 9: Convert Trillion Number*

* Input: 1000000000000.00
* Expected Output: "ONE TRILLION DOLLARS"

*Test Case 10: Convert Quadrillion Number*

* Input: 1000000000000000.00
* Expected Output: "ONE QUADRILLION DOLLARS"

*Test Case 11: Convert Zero*

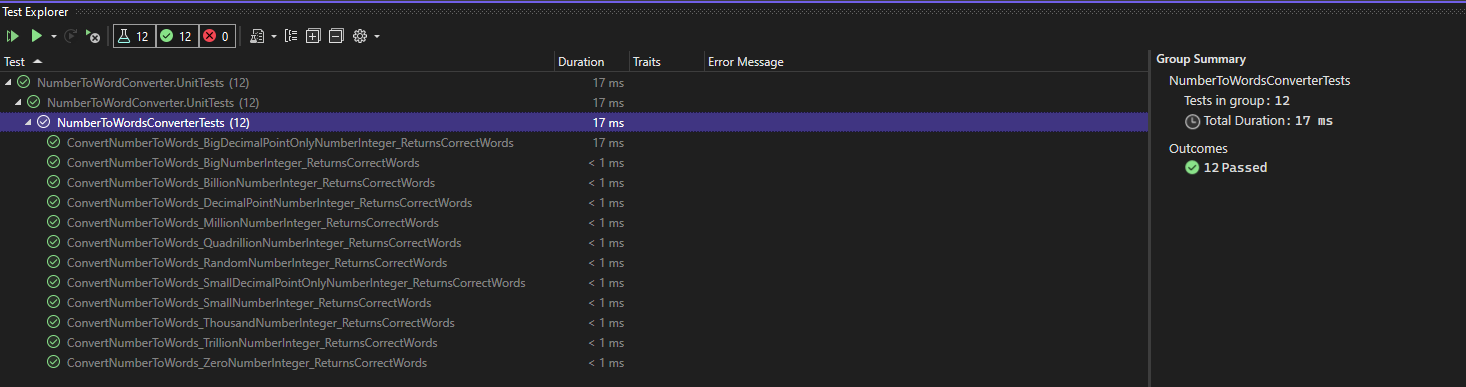
* Input: 00.00
* Expected Output: "ZERO DOLLARS"

*Test Case 12: Convert Random Number*

* Input: 590807.00
* Expected Output: "FIVE HUNDRED AND NINETY THOUSAND EIGHT HUNDRED AND SEVEN DOLLARS"

Test Execution:

|  |  |  |
| --- | --- | --- |
| Test Case | Pass | Fail |
| Test Case 1: Convert Small Number Integer | PASS |  |
| Test Case 2: Convert Big Number Integer | PASS |  |
| Test Case 3: Convert Small Decimal Number Integer | PASS |  |
| Test Case 4: Convert Big Decimal Number Integer | PASS |  |
| Test Case 5: Convert Decimal Number | PASS |  |
| Test Case 6: Convert Thousand Number | PASS |  |
| Test Case 7: Convert Million Number | PASS |  |
| Test Case 8: Convert Billion Number | PASS |  |
| Test Case 9: Convert Trillion Number | PASS |  |
| Test Case 10: Convert Quadrillion Number | PASS |  |
| Test Case 11: Convert Zero | PASS |  |
| Test Case 12: Convert Random Number | PASS |  |



Performance:

Overall performance is fast and acceptable, only big number integer take the longest time comparing to other test cases.

Conclusion:

The test results are good and meet the expectations result.