




Basic Addition Test

- **Description:** Verifies the addition of two integer numbers.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add 7 3`
- **Expected Result:** 10 OK 


Basic Subtraction Test

- **Description:** Verifies the subtraction of two integer numbers.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli subtract 10 4`
- **Expected Result:** 6 OK 


Basic Multiplication Test

- **Description:** Verifies the multiplication of two integer numbers.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 6 5`
- **Expected Result:** 30 OK 

Basic Division Test

- **Description:** Verifies the division of two integer numbers.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 20 4`
- **Expected Result:** 5 OK 


Division by Zero Test (Double-check of known issues)

- **Description:** Verifies the behavior when dividing by zero.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 10 0`
- **Expected Result:** Error message, "Cannot divide by zero." OK 

Decimal Subtraction Test

- **Description:** Verifies the behavior when subtracting with decimals.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli subtract 1.0001 0.0001`
- **Expected Result:** 1.0000. **NOK**
- **Note:** The result obtained is 1, which does not maintain the expected decimal precision. The calculator is rounding or truncating the result to an integer, indicating an issue with precision in decimal subtraction operations.


Decimal Multiplication Test

- **Description:** Verifies the behavior when multiplying with decimals.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 1.0001 0.0001`
- **Expected Result:** 0.00010001. OK 


Decimal Division Test

- **Description:** Verifies the behavior when dividing with decimals.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 1.0001 0.0001`
- **Expected Result:** 10001. **NOK**
- **Note:** The expected result should be 10001.0000 (or 10001 with 4 decimal precision). The calculator seems to round to an integer in this case, which could be an issue if decimal precision is required.


Scientific Notation Test

- **Description:** Verifies the result in scientific notation for large numbers.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 1e+10 1e+10`
- **Expected Result:** 1e+20 OK 

Decimal Accuracy Test

- **Description:** Verifies precision up to 8 decimal places.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add 1.00000001 1.00000001`
- **Expected Result:** 2.00000002 (should be checked if rounded to 2.0 as mentioned in known issues). OK 


Infinite Operation Test

- **Description:** Verifies the behavior for operations resulting in infinity.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 1e+100 1e-100`
- **Expected Result:** Infinity OK 

Negative Infinite Operation Test

- **Description:** Verifies the behavior for operations resulting in negative infinity.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide -1e+100 1e-100`
- **Expected Result:** -Infinity OK 

Negative Operands Test


- **Description:** Verifies the operation with negative operands.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add -5 -3`
- **Expected Result:** -8 OK 

Incorrect Number of Operands Test (Double-check of known issues)


- **Description:** Verifies the behavior when using more or less than two operands.

- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add 5 7 2`
- **Expected Result:** Error message, "Incorrect number of operands." **NOK**
- **Note:** According to known issues, the calculator should show an error message when more than two operands are provided. The fact that it returned 12 suggests that the calculator may be handling the number of operands incorrectly. The calculator should return an error message instead of attempting to perform the operation with more than two operands. This indicates a possible defect in error handling for this case.

Performance Tests

- **Description:** Verifies the performance of the calculator.
- **Command:** `for i in {1..1000}; do docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add $i $i; done`
- **Expected Result:** Sum of each value in i and should end at 2000 OK 

Stress Tests

- **Description:** Verifies the behavior with extreme operations.
- **Command:** `docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 9999999999999999 9999999999999999`
- **Expected Result:** Should work without any issues OK 

Additional Notes:

- **Confirmed Known Issues:**
 - Dividing by zero correctly displays an error message.
 - The calculator handles operations resulting in infinity, negative infinity, or NaN as expected.
 - Results are displayed in scientific notation when appropriate, and results are adjusted to 8 decimal places as expected.
- **Recommendations:**
 - Review precision in decimal operations.

- Improve error handling for the case of incorrect number of operands.

Automation

I decided to implement a shell script covering the cases mentioned above. To execute it, follow these steps:

1. Docker must be installed as specified in the project.
2. The image must be downloaded.
3. Download the attached `test_calculator.sh` file from the email.
4. Grant execution permission: `chmod +x test_calculator.sh` (must be in the directory where the file is located).
5. Run the script: `./test_calculator.sh`