Program Structure and Development

Mingzheng Michael Huang

- ***Command-Line Parsing:*** Processes command-line arguments to set program parameters, including total and mantissa bits, and operation type. Employs error checks and validations using if statements for robust input handling.
- ***Core Functions: 1.*** **convert**: Translates binary strings to floating-point numbers, involving loops for exponent and mantissa calculations and conditions for special cases like Infinity and NaN. **2. minmax**: Calculates minimum and maximum representable floating-point values based on user-defined bit specifications. **3. addHex**: Performs addition of two hexadecimal numbers by converting them to the custom floating-point format, integrating string manipulation with arithmetic operations.
- ***Helper Functions:*** Consists of conversion functions like **hexToBinary**, **binaryToFloat**, **floatToBinary**, and **binaryToHex**, each tailored for specific format transformations. These functions enhance code modularity and are implemented using character-wise processing loops.
- ***Main Function:*** Acts as the control center, directing the program based on user input to the appropriate function among convert, minmax, or addHex. It manages the execution flow and result presentation, ensuring cohesive operation of the program components.

Automated Validation with Bash Script - Overview

Script Functionality:

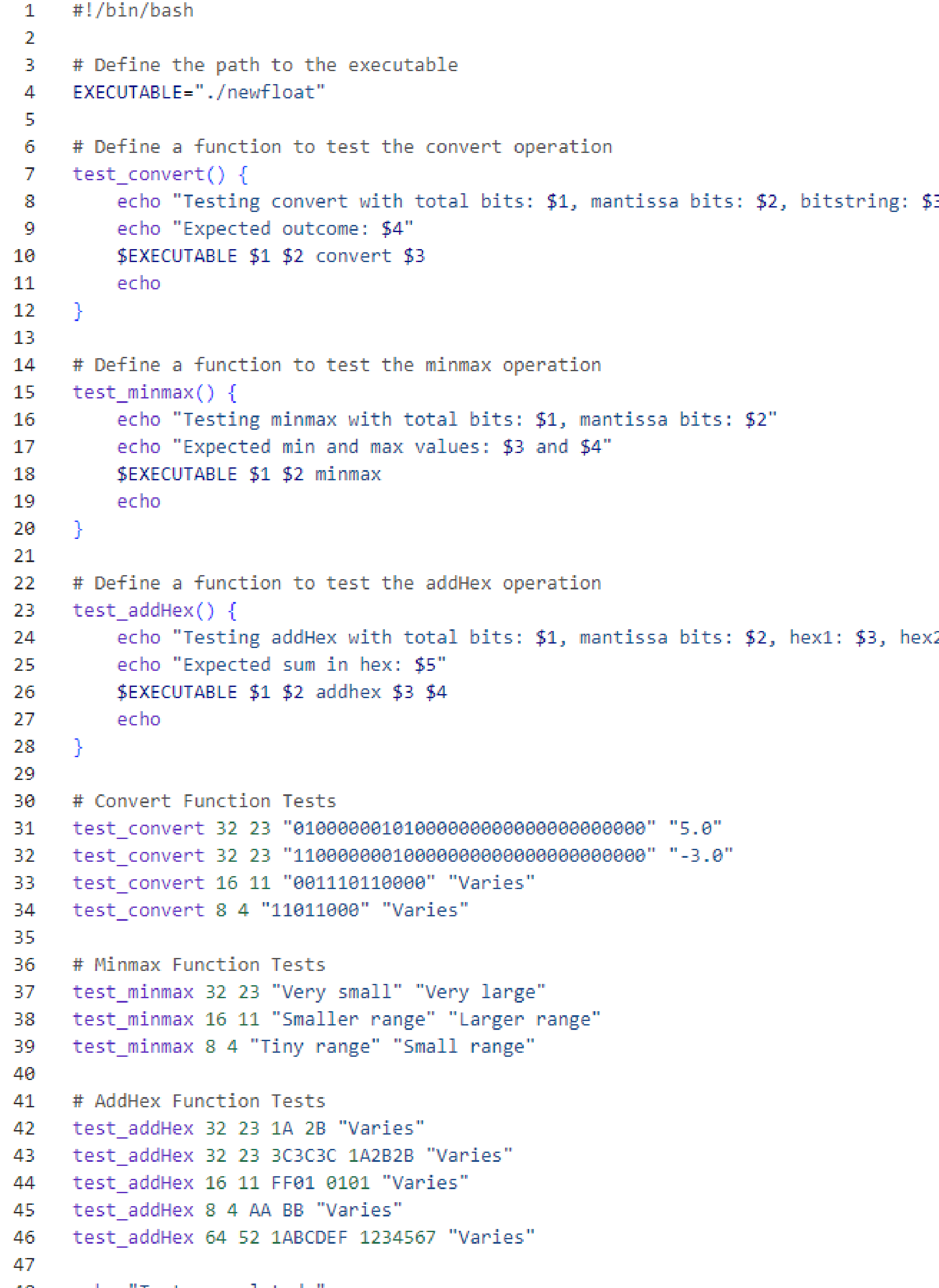
- Automated script leverages shell scripting to interact with ***newfloat*** executable, methodically testing ***convert***, ***minmax***, and ***addHex*** functions.
- Employs command-line argument passing to simulate user inputs for diverse floating-point operations.

Efficiency in Testing:

- Utilizes batch processing and automated execution to efficiently validate multiple scenarios, enhancing test throughput.
- Scripting allows for rapid iteration and regression testing, crucial for agile development cycles.

Precision of Evaluation:

- Script meticulously assesses floating-point precision, exponent handling, and mantissa computation accuracy.
- Ensures compliance with IEEE floating-point standard nuances, especially in convert function’s base-10 representation and ***addHex*** function's hexadecimal arithmetic.



```bash
#!/bin/bash

# Define the path to the executable
EXECUTABLE="./newfloat"

# Define a function to test the convert operation
test_convert() {
    echo "Testing convert with total bits: $1, mantissa bits: $2, bitstring: $3
    echo "Expected outcome: $4"
    $EXECUTABLE $1 $2 convert $3
    echo
}

# Define a function to test the minmax operation
test_minmax() {
    echo "Testing minmax with total bits: $1, mantissa bits: $2"
    echo "Expected min and max values: $3 and $4"
    $EXECUTABLE $1 $2 minmax
    echo
}

# Define a function to test the addHex operation
test_addHex() {
    echo "Testing addHex with total bits: $1, mantissa bits: $2, hex1: $3, hex2
    echo "Expected sum in hex: $5"
    $EXECUTABLE $1 $2 addhex $3 $4
    echo
}

# Convert Function Tests
test_convert 32 23 "01000000101000000000000000000000" "5.0"
test_convert 32 23 "11000000010000000000000000000000" "-3.0"
test_convert 16 11 "001110110000" "Varies"
test_convert 8 4 "11011000" "Varies"

# Minmax Function Tests
test_minmax 32 23 "Very small" "Very large"
test_minmax 16 11 "Smaller range" "Larger range"
test_minmax 8 4 "Tiny range" "Small range"

# AddHex Function Tests
test_addHex 32 23 1A 2B "Varies"
test_addHex 32 23 3C3C3C 1A2B2B "Varies"
test_addHex 16 11 FF01 0101 "Varies"
test_addHex 8 4 AA BB "Varies"
test_addHex 64 52 1ABCDEF 1234567 "Varies"

```

Automated Validation with Bash Script - Execution and Results

Execution Process:

- Sequential execution of predefined test cases, invoking newfloat with parameters like total bits, mantissa bits, and operation-specific inputs.
- Automated parsing of output to validate against expected floating-point representation and arithmetic results.

Case-by-Case Validation:

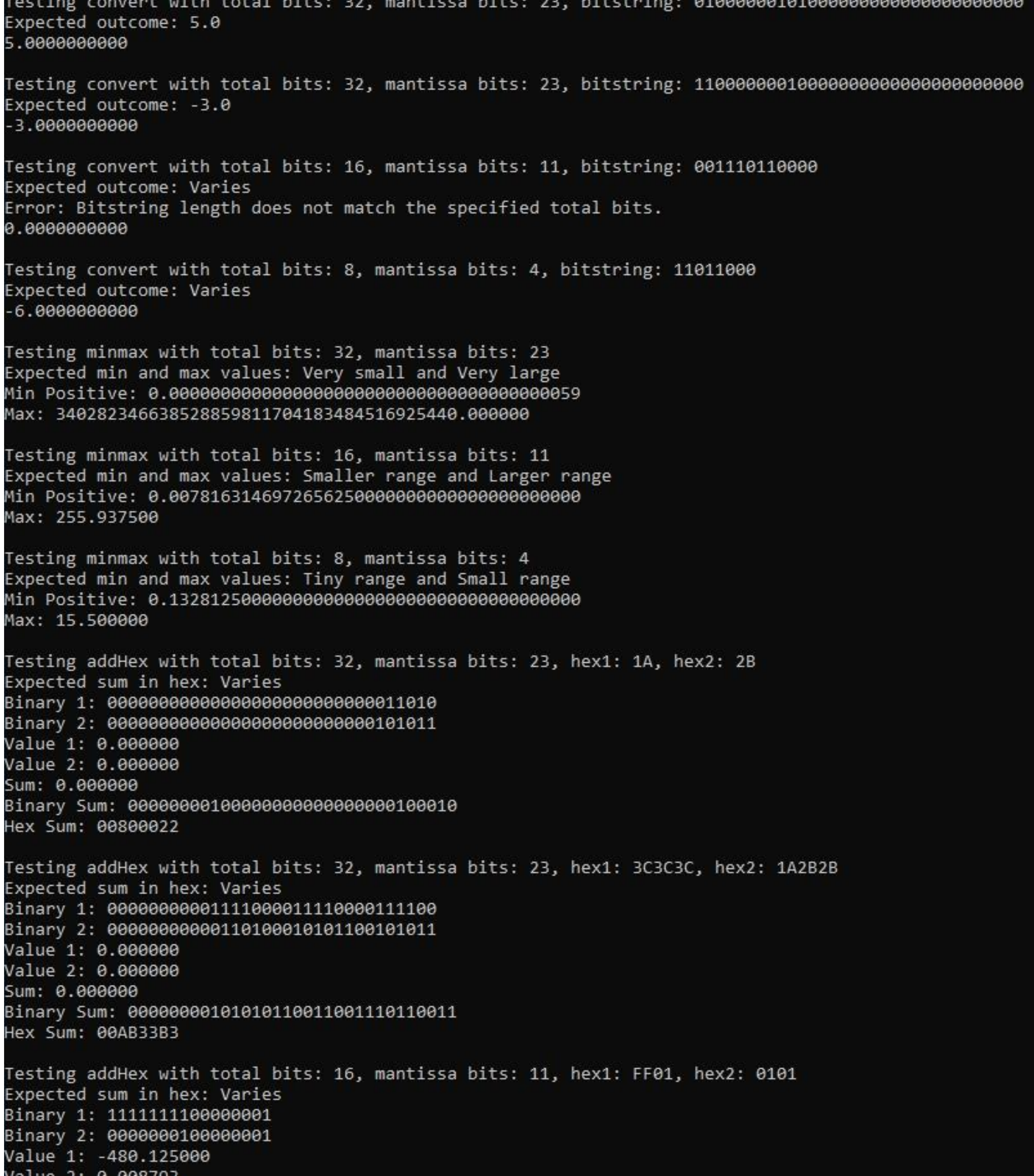
- Detailed breakdown of test cases, for example:

1. ***convert*** function testing with varied bit patterns to ensure accurate IEEE floating-point representation.
2. ***minmax*** function validation against theoretical minimum and maximum floating-point values.
3. ***addHex*** function's hexadecimal arithmetic tests, focusing on precision and correctness in binary-to-float conversion.

Emphasizes script's role in validating numerical accuracy and edge-case handling in floating-point calculations.

Outcome and Insights:

- Summarizes key findings such as precision accuracy in floating-point operations, adherence to specified bit formats, and robustness in arithmetic operations.
- Insights on script's effectiveness in identifying precision-related issues and compliance with floating-point standards.



```
Testing convert with total bits: 32, mantissa bits: 23, bitstring: 01000000101000000000000000000000
Expected outcome: 5.0
5.0000000000

Testing convert with total bits: 32, mantissa bits: 23, bitstring: 11000000010000000000000000000000
Expected outcome: -3.0
-3.0000000000

Testing convert with total bits: 16, mantissa bits: 11, bitstring: 001110110000
Expected outcome: Varies
Error: Bitstring length does not match the specified total bits.
0.0000000000

Testing convert with total bits: 8, mantissa bits: 4, bitstring: 11011000
Expected outcome: Varies
-6.0000000000

Testing minmax with total bits: 32, mantissa bits: 23
Expected min and max values: Very small and Very large
Min Positive: 0.0000000000000000000000000000000000000059
Max: 340282346638528859811704183484516925440.000000

Testing minmax with total bits: 16, mantissa bits: 11
Expected min and max values: Smaller range and Larger range
Min Positive: 0.0078163146972656250000000000000000000000
Max: 255.937500

Testing minmax with total bits: 8, mantissa bits: 4
Expected min and max values: Tiny range and Small range
Min Positive: 0.1328125000000000000000000000000000000000
Max: 15.500000

Testing addHex with total bits: 32, mantissa bits: 23, hex1: 1A, hex2: 2B
Expected sum in hex: Varies
Binary 1: 00000000000000000000000000011010
Binary 2: 00000000000000000000000000101011
Value 1: 0.000000
Value 2: 0.000000
Sum: 0.000000
Binary Sum: 00000000100000000000000000100010
Hex Sum: 00800022

Testing addHex with total bits: 32, mantissa bits: 23, hex1: 3C3C3C, hex2: 1A2B2B
Expected sum in hex: Varies
Binary 1: 00000000001111000011110000111100
Binary 2: 00000000000110100010101100101011
Value 1: 0.000000
Value 2: 0.000000
Sum: 0.000000
Binary Sum: 00000000101010110011001110110011
Hex Sum: 00AB33B3

Testing addHex with total bits: 16, mantissa bits: 11, hex1: FF01, hex2: 0101
Expected sum in hex: Varies
Binary 1: 1111111100000001
Binary 2: 0000000100000001
Value 1: -480.125000
Value 2: 0.008703
```