SOEN 6011: Project Team C Student N4

Kenlo DINH VAN (26641652)

1 Function Description

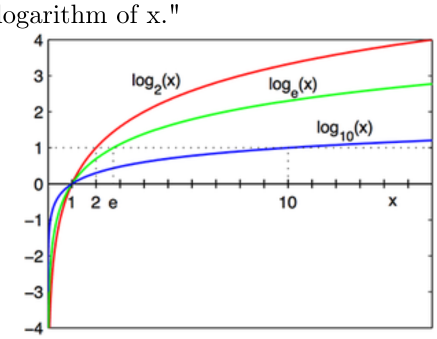
1.1 Logarithmic Function

$$\log_b(x)$$

The **logarithmic** function is the **inverse** of the exponential function, since it is a one-to-one function. The graph of an inverse function is the reflection of the original function, using the line y = x as reflection axis. John Napier expressed y as a function of x for the logarithm in 1614 resulting in:

$$\log_b(x) = y$$

which can be read: "x is equal to b (base) to the power y", which is equivalent to "y is the base-b logarithm of x."



- **Domain:** x > 0 (set of positive real numbers)
- **Co-domain**: Set of real numbers **R**
- **Specificity of the base**: b $\neq$ 1 and b > 0

1.2 Unique characteristics

Exponential expressions can be written as logarithmic expressions and logarithmic expressions can be written as exponential expressions. ex: 3^2=9 => $\log_3(9)$ = 2

- when b = 10 the function is called common logarithm and is denoted log(x)
- when b = e = 2.718... function is called natural logarithm and denoted ln(x)

Change of base

$\log_b x = \frac{\log_k x}{\log_k b}$.

Logarithmic identities

- Product: $\log_b(x * y) = \log_b x + \log_b y$
- Quotient: $\log_b(x/y) = \log_b x - \log_b y$
- Power: $\log_b(x^p) = p \log_b x$
- Root: $\log_b \sqrt[p]{x} = \frac{\log_b x}{p}$

2. Assumptions

Assumption 1

As of version 0, we assume that the user interface will be text based relying on console input-output.

Assumption 2

The 'system' refers to the scientific calculator, Eternity: Functions

3. Functional requirements

F4.0.v1 Function Selection

When the system starts, the interface should display the function name and allow the user to select the logarithmic function. **Priority: Low, Risk: Low, Difficulty: Low**

F4.1.v1 Logarithm Base Initialization

When the user selects the logarithmic function, the system should ask the user which base b he wishes to use and set the base value. **Priority: Medium, Risk: Low, Difficulty: Low**

F4.2.v1 Logarithm Base Validation

After the user inputs the base value b, the system should validate that b is a real positive number not equal to 1. **Priority: Medium, Risk: Low, Difficulty: Medium**

F4.3.v1 Logarithm Variable Initialization

If the base b is valid, the system should ask the user to input the value for variable x and set it. **Priority: Medium, Risk: Low, Difficulty: Low**

F4.4.v1 Logarithm Variable Validation

After the user inputs the variable x value, the system should validate that the variable x is a real positive number. **Priority: Medium, Risk: Low, Difficulty: Medium**

F4.5.v1 Logarithm Calculation

If the variable x is valid, the system should calculate the logarithm of x in base *b,* without relying on java built-in functions, and store the result. **Priority: High, Risk: High, Difficulty: High**

F4.6.v1 Result Display

After the calculation completes, the system should display the result on the user interface. **Priority: Low, Risk: Low, Difficulty: Low**

4. Considered Algorithms

4.1 Algorithm 1: Binary Logarithm Approximation

The binary logarithm may be computed in two parts: an integer part (characteristic) and a fractional part (mantissa of the logarithm).

$$\log_2 x = n + \log_2 y \quad \text{where } y = 2^{-n}x \text{ and } y \in [1, 2)$$

- **Advantages:** Mathematically sound, potential for a very precise result due to this 2 parts binary log value approximation. (set of positive real numbers)
- **Disadvantages**: Complex. For practicality, this infinite series must be truncated to reach an approximate result.

Algorithm 1 Binary Logarithm Approximation

```
function Logarithm(x, base)
    r ← Binary Log(x)/Binary Log(base)
    return r
end function

function Binary Log(x)
    n ← Binary Log Integer(x)
    y ← x/2^n
    f ← Binary Log Fraction(y)
    return b ← n + f                ▷ returns the integer and fractional parts
end function

function Binary Log Integer(x)                ▷ returns 'p' a power of 2
    p ← 0
    v ← 1
    while v ≤ x do
        v ← v * 2
        if v < x then
            p ← p + 1
        end if
    end while
    return p
end function

function Binary Log Fraction(y)
    if y == 1 then
        return 0
    end if
    z ← y
    m ← 0
    while z < 2 do                ▷ square until z in [2;4)]
        z ← z^2
        m ← m + 1
    end while
    return log_2 y ← 2^(−m) + 2^(−m) Binary Log Fraction(z/2)
end function
```

4.2 Algorithm 2: Natural Algorithm Approximation

The second algorithm will rely on approximating the natural logarithm by considering the following identity:

$$\ln(x) \;=\; \lim_{n\to\infty} \; n \; (\; x^{1/n} \; - \; 1)$$

- **Advantages**: Simpler to implement.
- **Disadvantages**: Possible lack of precision.

Algorithm 2 Natural Logarithm Approximation

---

function LOGARITHM(x, base)
    $r \leftarrow$ NATURAL LOG(x)/NATURAL LOG(base)
    return r
end function

function NATURAL LOG(x)
    $a \leftarrow 1000$
    return $n \times (x^{(1/n)} - 1)$
end function

References

[1] https://en.wikipedia.org/wiki/Logarithm
[2] https://en.wikipedia.org/wiki/Binary_logarithm
[3] http://dwb4.unl.edu/Chem/CHEM869R/CHEM869RMats/Logs/Logs.html