SOEN 6011 : SOFTWARE ENGINEERING PROCESSES

SUMMER 2022

ETERNITY

Instructor: PANKAJ KAMTHAN

Project Report

Student Name: Yun Ni
Student ID: 40179775
July, 2022

Overleaf:`https://www.overleaf.com/read/tsccdnnpvqpw`
Github: `https://github.com/ninanee/SOEN6011_Project`

Contents

1 Introduction

The function $f(x) = ab^x$ is an exponential function where $a \neq 0$ and $b > 0$. In this exponential function, a is the initial quality, b is the growth factor or decay factor and c is the exponent. The function is called exponential because it has the input variable in the exponent and we can repeatedly multiplying by b [2].

1.1 Domain

The domain of this function is given by all the values that x can take. If b is a positive real number and $a \neq 0$, x can take any value. Therefore, the domain is $x \in R$.

1.2 Co-domain

The co-domain is given by the dependent values of the variable y, where $y = ab^x$. If we evaluate $f(-\infty)$ then y tends to zero. Besides, if we evaluate $f(\infty)$ then y tends to infinity[10]. Therefore, the co-domain is $(0, \infty)$.

1.3 Characteristics

- Growth: if $b > 1$, the function represents exponential growth which is a function increasing at a constant percentage. It is shown on the left hand side of Figure 1.
- Decay: if $0 < b < 1$, the function represents exponential decay which is a function decreasing at a constant percentage. It is shown on the right hand side of Figure 1.
- Invective and Subjective: this function is subjective since for every y, there is an x such that $f(x) = y$, and it is not an invective function.

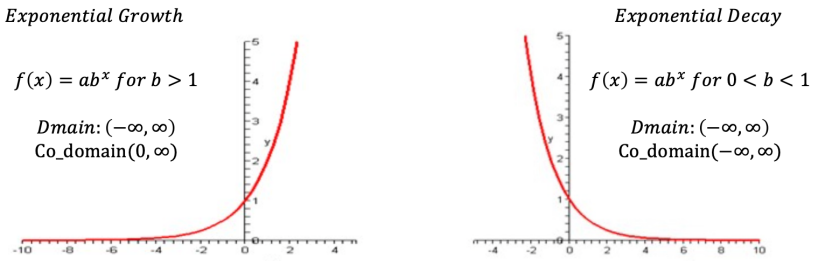

Figure 1: The graph representation of $f(x) = ab^x$

1.4 Context of Use Model

The context of use model for the exponential function calculator[12][7] is represented as Figure 2.

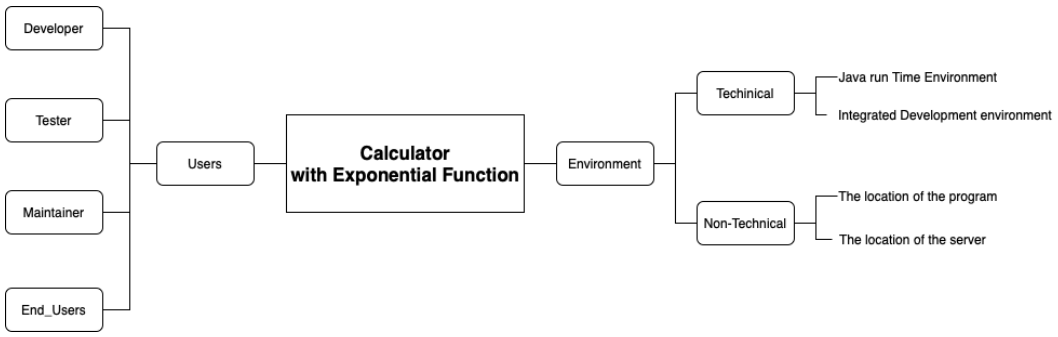

Figure 2: The Context Diagram for exponential function calculator

1. End User: Users who use the calculator to get the output of ab^x with the input x.
2. Task: Calculate the output of ab^x with the input x and show the result to the user.
3. Environment:
   - Technical environment: The power function of the used calculator. This calculator can't be used without power function.
   - Non-technical environment: The location of where the user use this calculator.

2 Project Requirements

This section presents the assumptions and requirements[5] for implementing $f(x) = ab^x$.

2.1 Assumptions

1. The Function will accept only real numbers as inputs.
2. The Function will be able to handle entry value of double-precision floating-point.

2.2 Requirements

Identification	FR1
Type	Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Medium
Description	System needs to take inputs x, a, b to give an output of ab^x. Besides, needs to define the constrains that $a \neq 0, b > 0$. Because, when $a = 0$ the function will be simplified to $y = f(x) = 0$.

Identification	FR2
Type	Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Medium
Description	Function ab^x does not depend on any other functions like built-in or library functions.

Identification	FR3
Type	Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Medium
Description	Users can give an input from all real numbers for x.

Identification	FR4
Type	Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Easy
Description	When users give other inputs than a number like a string, the system should not accept and should show the exception properly.

Identification	FR5
Type	Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Easy
Description	When users give inputs does not specify all there inputs like a, b, x, the system should not accept and throw an error.

Identification	FR6
Type	Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Easy
Description	Ensure the accuracy of decimal operations to the power of decimals.

Identification	NFR1
Type	Non-Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Easy
Description	Users need to have the Java environment to use the project.

Identification	NFR2
Type	Non-Functional Requirement
Version No.	1.0
Owner	Yun Ni
Priority	High
Difficulty	Easy
Description	The program should be modular and can be tested.

3 Algorithm

This section presents the pseudo-code and algorithm for implementing $f(x) = ab^x$.

3.1 Mind map

The mind map for deciding a pseudocode format and algorithm design can be checked with Figure3 and Figure4

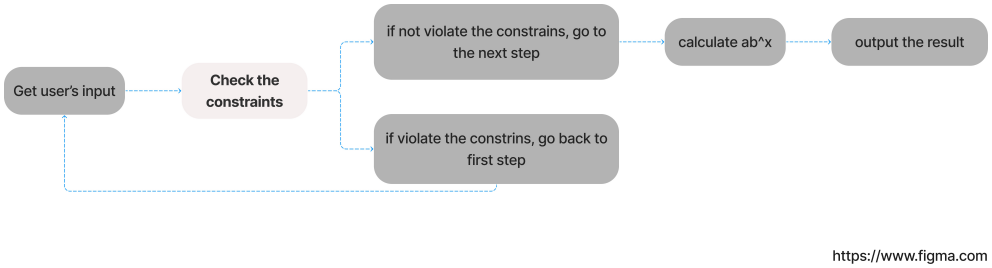

Figure 3: The mind map of algorithm design

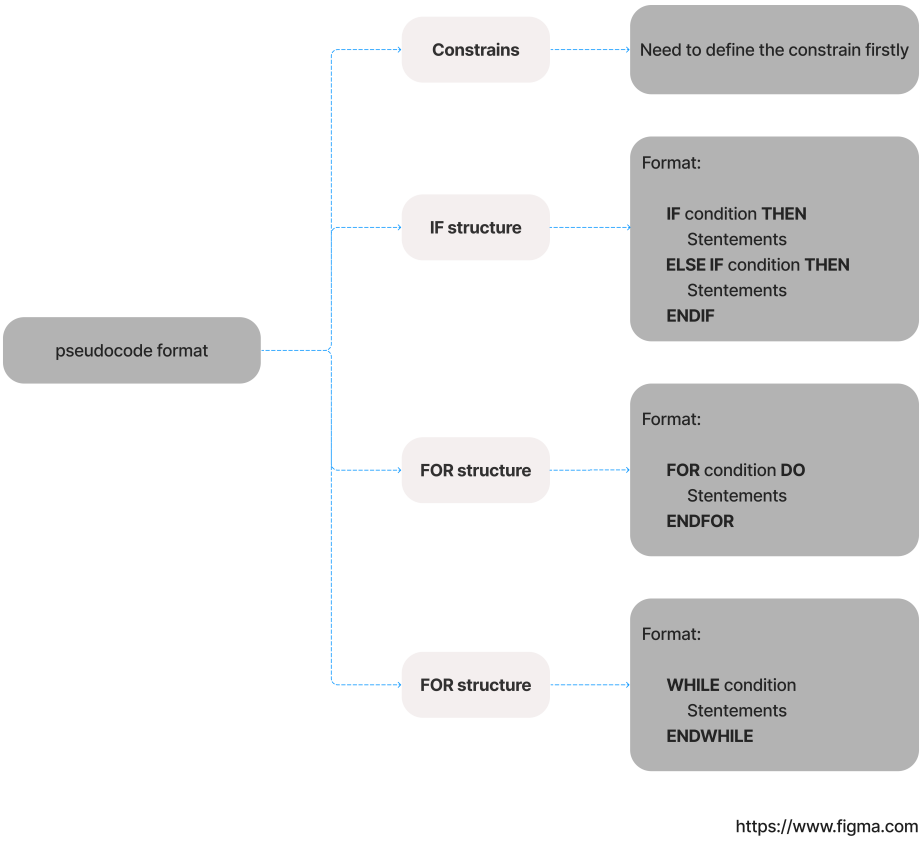

Figure 4: The mind map of pseudocode format

3.2 Algorithm 1: Iterative

The interactive algorithm can execute the steps in number of iterations, using the loop. We can track the result in a variable called prod, which is initially set to 1 and then use loop from 1 to x number of times. By incriminating by one on each iteration, we multiply prod by b. At the end of the loop value of prod is equal to b^x then return the result of $a * prod$[11].

Algorithm 1 Iterative algorithm for calculating $f(x) = ab^x$

Require: $a \neq 0$ and $b > 0$

```
1: function (exponent_iterative(a,b,x))
2: in: double number x, a, b
3: out: double number prod
4:     prod ← 1
5:     temp ← 1
6:     for temp ≤ x do
7:         prod ← prod * b
8:         temp ← temp + 1
9:     end for
10:    prod ← prod * a
11:    return prod
12: end function
```

Advantages	**Disadvantages**
Easy to implement, since it uses looping structure.	If the condition fails, the iteration terminates the execution of the program.
Tracing the iterative algorithm is easy.	It is not very efficient in terms of time since the time complexity is $O(n)$ for a larger input[11].
Could avoid memory overflow of input.	Can not calculate decimal inputs properly in this case.

3.3 Algorithm 2: Recursive

A recursive function calls by itself during the execution of the program, until the base conditions are met[3]. By defining a helper function called exponent_helper which calculates b^x in different conditions. With the helper function, we multiply the result to the value of input a and return it to the user.

Algorithm 2 Recursive algorithm for calculating $f(x) = ab^x$

Require: $a \neq 0$ and $b > 0$

```
1: function (exponent_recursive(a,b,x))
2: in: double number x, a, b
3: out: double number prod
4:     power ← exponent_helper(b, x)
5:     prod ← power * a
6:     return prod
7: end function
```

```
1: function (exponent_helper(b,x))
2: in: number x, b
3: out: double number prod
4:     if x < 0 then
5:         b ← 1.0/b
6:         x ← −x
7:         return exponent_helper(b, x)
8:     else if x = 0 then
9:         return 1.0
10:    else if  x = 1 then
11:        return b
12:    else if x  mod 2 = 0 then
13:        b ← b ∗ b
14:        x ← x/2
15:        return exponent_helper(b, x)
16:    else
17:        b ← b ∗ b
18:        x ← x − 1
19:        x ← x/2
20:        return exponent_helper(b, x)
21:    end if
22: end function
```

Advantages	Disadvantages
The time complexity of the defined recursive algorithms above is $O(\log n)$.	Recursive algorithm is not easy to debug since it calls itself.
Recursive algorithms is easier to maintain than loop[6].	Recursive algorithm needs the system continuously allocates memory, might lead stack overflow[3]
	Can not calculate decimal inputs properly in this case.

3.4 Algorithm 3: Taylor Series for calculating $f(x) = ab^x$

For calculating the exponential function $f(x) = ab^x$ more properly, we can use Taylor Series algorithm. When x is a decimal, Taylor Series can calculate $\ln(x)$ and e^x with a high precision, since the formula can be changed to $b^x = e^{b \ln b}$. When x is an integer, if x is an even number, b^x can be decomposed to $(b^2)(x/2)$; if x is an odd number, b^x can be decomposed to $b \cdot ((b^2)((x-1)/2)$. And then continue to decompose until the exponent part is equal to one[1].

Algorithm 3 Exponentiation by Taylor Series[8]

Require: $a \neq 0$ and $b > 0$

```
function LOGARITHM(n)
    sum ← 0
    while n > 1 do
        n ← n/e                    ▷ e is a constant approximately equal to 2.71828
        x ← x + 1
    end while
return x
end function
function EXPONENTIAL(b)
    sum ← 1
    n ← 10
    for i ← n − 1, 1 do
        sum ← 1 + b ∗ sum/i
    end for
return sum
end function
log b ←LOGARITHM(b)
result ←EXPONENTIAL(x ∗ log b)
result ←a·(result)
```

Advantages	Disadvantages
The Taylor Series is very useful for derivations and can calculate even the most complex functions.	The algorithm could be very complex with deriving.
The Taylor Series can be used to get theoretical error bounds and estimate the value of any input[4].	The expansion point might lead truncation errors.

3.5 Algorithm Choice

For a function, the accuracy is the most significant factor, since a wrong output is worse than a blank. The iterative and recursion algorithms can not calculate the decimal part of $f(x) = ab^x$ with high precision, therefore, my decision is to use Taylor Series algorithm and the sub-function of the recursive algorithm to implement the project. It runs more efficiently and provide results with high accuracy.

4 Code Implementation

4.1 Debugger

IntelliJ has a standard build-in debugger which allows the program to open in debug mode. It supports both step by step debugging, break point, check points and multiple views which enhance the experience of debugging.

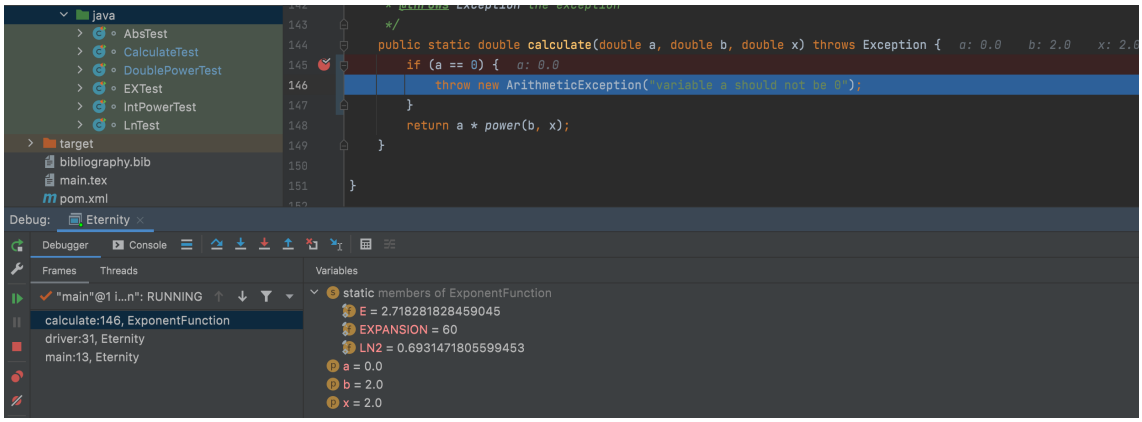

Figure 5: The snapshot of the debugger in IntelliJ

4.1.1 Advantages of the Debugger

- As a build-in tool, it can be easily accessed by users. Users can create various kinds of breakpoints, such as conditional breakpoints to trace the source code.

- Users can stop execution at a given point to investigate where it goes and what the values are.
- The hot swapping function allows users to insert minor changes in the code without shutting down the process which is convenient.[9]

4.1.2 Disadvantages of the Debugger

- For the multi-threading programs, the debugger tool in IntelliJ could not work efficient.
- The debugger can not solve the bad code or correct the code.
- The IntelliJ may run into bad state and have to clear the caches and restart the IDE to go through the build-clean-build to fix it.

4.2 The Quality Attributes

The quality attributes assessed while implementing the source code and program are:

- **Efficiency**: In order to achieve efficiency, the primitive data types are used instead of using the user-defined data type. Therefore, the output is generated in the order of milliseconds.
- **Portability**: With the executable jar file, users can use the program without any difficulties which could increase the portability.
- **Maintainability**: Giving the simple and understandable names to variables could improve the maintainability. If there is an error in the function, I can locate the errors by their names. Besides, the core logic and the error handling have been properly implemented with comments.
- **Correctness**: For the function $y = ab^x$, when x is an integer, the result can get easily by a loop or a recursion. When x is a decimal, the loop could generate incorrect result. To improve the accuracy, the Taylor Expansion is used to get the similar value of the b^x, then multiply by a which could reduce the error.
- **Robustness**: My program supports for the error handling, exceptions and thrown to handle errors. Therefore, the program will not crash and notify the users with appropriate error messages.
- **Usability**: By providing the textual user interface for the project, users can use the calculator easily.

4.3 Checkstyle

CheckStyle for java can scan the source code and point out coding issues that the code might have missed or might lead to problems. It can also help improve the Java Coding because we have to investigate and decide if this is a problem or not. I am using Google Checkstyle in the program like in Figure6 and Figure7.

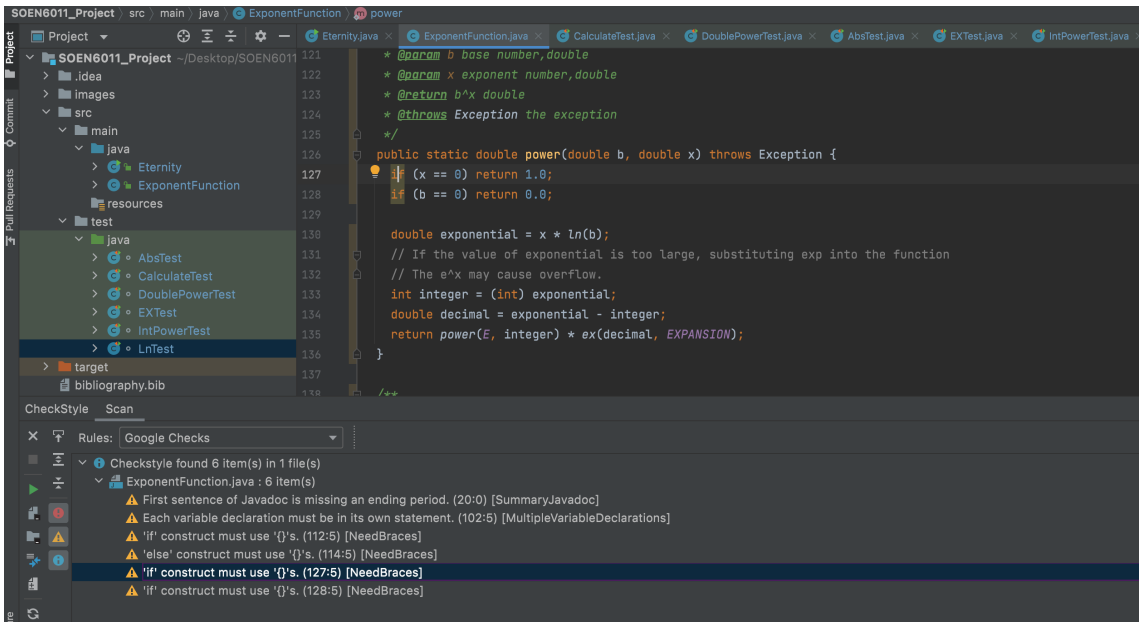

Figure 6: The snapshots of the Checkstyle

4.3.1 Advantages of the Checkstyle

- It is a convenient external plug and can be installed directly in the IntelliJ IDEA.
- It can check the layout and the formatting issue.
- It gives developers the ability to customize their own rules for checking.

4.3.2 Disadvantages of the Checkstyle

- It is limited to only the code beautification. No logic checks at present.
- There is no modification function after checking. Developers need to modify the problems one by one.

5 Unit Test

Junit is a unit testing framework for Java programming language and has been import to the development of test-driven developments. In this project, Junit4 has been used to write

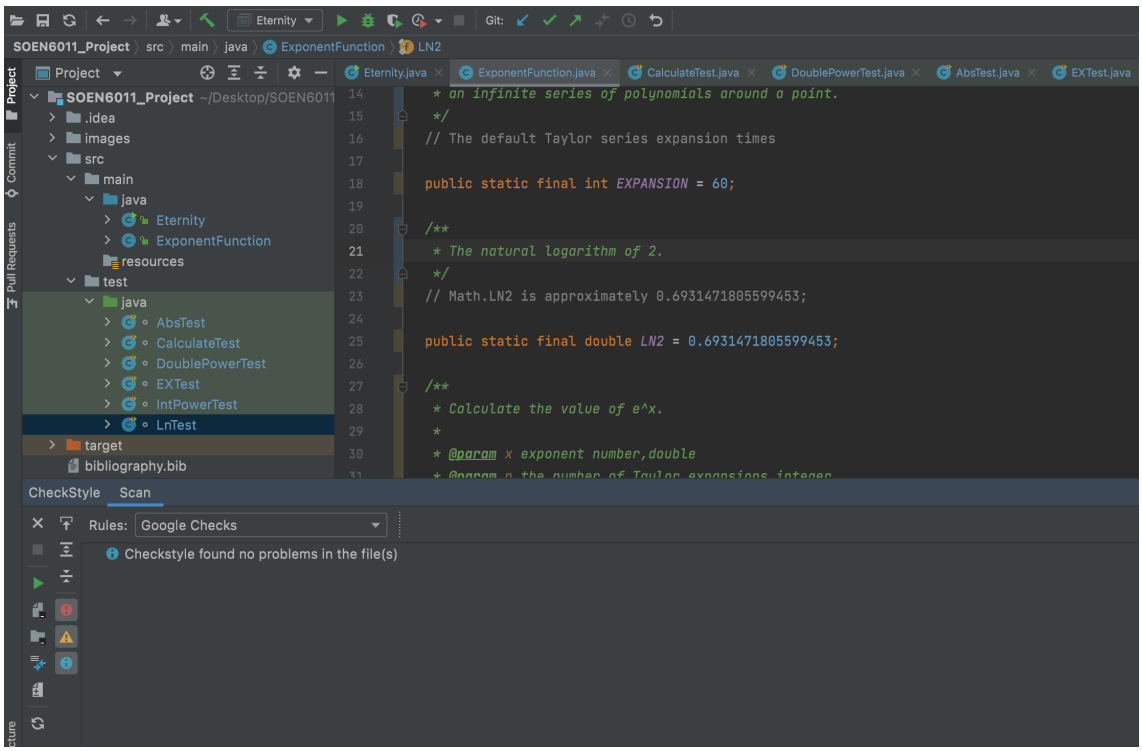

Figure 7: The snapshots of the Checkstyle

the test cases.

5.1 Mapping Requirements with Test Cases

Requirement Identifier	Test Cases
FR1	exceptionCalculate(), negativeCalculate()
FR2	absZero(), lnZero(), zeroPowerZero()
FR3	zeroPowerZero(), zeroPowerRationalNumber()
FR4	lnOfNegativeValues(), lnZero()
FR5	ePowerRealNumber(), exceptionCalculate()
FR6	positiveNumberPowerZero(), positiveNumberPowerOne()

5.2 Test Cases

Test Case ID	Test Case 1
Class Name	AbsTest
Method Name	absZero(), absPositive(), absNegative()
Description	Testing the absolute value function
Input(s)	0; 5; -5
Expected Output(s)	0; 5; 5
Actual Output(s)	0; 5; 5
Test Result	Passed

Test Case ID	Test Case 2
Class Name	EXTest
Method Name	ePowerZero(), ePowerPositive(), ePowerNegative(), ePowerRealNumber()
Description	Testing e^x function with expansion equals 60
Input(s)	0; 10; -10; 2.71818
Expected Output(s)	1; 22026.46; 4.53; 15.15
Actual Output(s)	0; 5; 5
Test Result	Passed

Test Case ID	Test Case 3
Class Name	LnTest
Method Name	lnZero(), lnOfPositiveValues(), lnOfNegativeValues()
Description	Testing ln^x function
Input(s)	0; 1; 53; -2
Expected Output(s)	throw exception; 0; 3.97; throw exception
Actual Output(s)	throw exception; 0; 3.97; throw exception
Test Result	Passed

Test Case ID	Test Case 4
Class Name	IntPowerTest
Method Name	zeroPowerZero(), positiveNumberPowerZero(), negativeNumberPowerPositiveEvenNumber(), negativeNumberPowerNegativeNumber(), positiveNumberPowerOne()
Description	Testing b^x function with input with integers
Input(s)	(0, 0); (-4, 0); (-3, 4); (-5, -9); (7, 1)
Expected Output(s)	1; 1; 81; -5.12E-7; 7
Actual Output(s)	1; 1; 81; -5.12E-7; 7
Test Result	Passed

Test Case ID	Test Case 5
Class Name	DoublePowerTest
Method Name	zeroPowerZero(), positiveNumberPowerZero(), negativeNumberPowerNegativeNumber(), positiveNumberPowerZero()
Description	Testing b^x function with inputs type double
Input(s)	(0.0, 0.0); (7.0, 1.0); (5.5, -9.6); (0.0, 5.8)
Expected Output(s)	1.0; 6.99; throw error; 0.0
Actual Output(s)	1.0; 6.99; throw error; 0.0
Test Result	Passed

Test Case ID	Test Case 6
Class Name	CalculateTest
Method Name	integerCalculate(); positiveCalculate(); exceptionCalculate(); positiveCalculate()
Description	Testing ab^x function with inputs
Input(s)	(5, 3, 10); (2, 3.6, -2.3); (0, 2, 2); (6.3, 1.2, 1.3)
Expected Output(s)	295245; 0.10508343093; throw exception; 8.36526215004
Actual Output(s)	295245; 0.10508343093; throw exception; 8.36526215004
Test Result	Passed

6 Conclusion

Building a project, it is necessary to have an explicit scoping and separate each functional module. With this project, I understand the process of each stage. Even this project is a small scale, I met each stage of the project development process and gained the knowledge with the various implementation methods and tools which can be used officially in the real industry.

References

[1] Alberto Abad, Roberto Barrio, Fernando Blesa, and Marcos Rodríguez. Algorithm 924: Tides, a taylor series integrator for differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 39(1):1–28, 2012.

[2] Andrew Browder. *Mathematical analysis: an introduction*. Springer Science & Business Media, 2012.

[3] Bruria Haberman and Haim Averbuch. The case of base cases: why are they so difficult to recognize? student difficulties with recursion. In *Proceedings of the 7th annual conference on innovation and technology in computer science education*, pages 84–88, 2002.

[4] G Hariharan and K Kannan. A comparative study of a haar wavelet method and a restrictive taylor's series method for solving convection-diffusion equations. *International Journal for Computational Methods in Engineering Science and Mechanics*, 11(4):173–184, 2010.

[5] IEC ISO. Ieee 12207-2008-iso/iec/ieee international standard-systems and software engineering–software life cycle processes, 2018.

[6] G Karigl. A recursive algorithm for the calculation of identity coefficients. *Annals of human genetics*, 45(3):299–305, 1981.

[7] Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. *Advances in neural information processing systems*, 14, 2001.

[8] Stack Overflow. Efficient generation of taylor (maclaurin) series. `https://stackoverflow.com/questions/23821878/efficient-generation-of-taylor-maclaurin-series`, 07 2022.

[9] StackExchange. What's the benefit of avoiding the use of a debugger? `https://softwareengineering.stackexchange.com/questions/131377/whats-the-benefit-of-avoiding-the-use-of-a-debugger`, 07 2022.

[10] MathBits Teacher. Exponential functions. `https://mathbitsnotebook.com/Algebra2/Exponential/EXExpFunctions.html`, 07 2022.

[11] Ling Xu. Application of the newton iteration algorithm to the parameter estimation for dynamical systems. *Journal of Computational and Applied Mathematics*, 288:33–43, 2015.

[12] M Zelen. Application of exponential models to problems in cancer research. *Journal of the Royal Statistical Society. Series A (General)*, pages 368–398, 1966.