Concordia University


**SOEN-6011 Software Engineering Processes**

---

# SCIENTIFIC CALCULATOR
## Function 9: Power Function
### DELIVERABLE 1 (D1)

---

***Author: Sanchit Kumar(40081187)***

July 19, 2019

# 1 Function 9 : Power Function

## 1.1 Description

A power function is of the form:

$$f(x) = x^y \tag{1}$$

where y is a real number.

## 1.2 Domain

1. When y is a non-negative integer, the domain is all real numbers: (- $\infty$,$\infty$)
2. When y is a negative integer, the domain is all real numbers excluding zero ( (- $\infty$, 0) $\cup$ (0,$\infty$) )
3. When y is a irrational number and y $>$ 0, the domain is all non-negative real numbers.
4. When y is a irrational number and y $<$ 0, the domain is all positive real numbers.

## 1.3 Characteristics of Power Function.

1. The behaviour of power function depends on whether the y is a positive or a negative number.
2. The behaviour of power function depends on whether the y is even or odd.
3. Also, the power function behaves differently for fractional powers and specifically for negative or positive fractional powers.

# 2 Requirements Specification

## 2.1 Definitions and abbrevations

Table 1: Definitions and abbreviations.

| Terms | Definition |
|---|---|
| FR | Funtional Requirement |
| NFR | Non-Functional Requirement |
| User | Someone who interacts with the system. |
| System | Software Program for calculation of Power Function. |

## 2.2 Constraints and Assumptions

1. User should provide input for both "x" and "y". No default values to be used.
2. Based on function characteristics, value of "x" and "y" should be a real number.
3. If input value for "x" is less than 0, then "y" is a whole number.
4. The output is contrained by Hardware.
5. The maximum value program could calculate is 3.40282346638528860e+38.
6. The minimum value program could calculate is -3.40282346638528860e+38.

# 3 Requirememts

## 3.1 Functional Requirements

- **ID** :FR1
  **TYPE** :Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :System should prompt the user to enter the value of x and y.
  **RATIONALE** :In order to get user input and start calculation.

- **ID** :FR2
  **TYPE** :Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :System should display an error message when value entered by user is not a number.
  **RATIONALE** :For calculations, input should be numbers only.

- **ID** :FR3
  **TYPE** :Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :In case the input entered is not valid system should prompt the user to input values again.
  **RATIONALE** :User should have the flexiblity to do calculations without exiting the program.

- **ID** :FR4
  **TYPE** :Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :User should have the option to exit the program anytime during

the use.
**RATIONALE** :If user is done with the use of program.

- **ID** :FR5
  **TYPE** :Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :System should display an error message when user enter value of x and y both as 0.
  **RATIONALE** :0 raised to the power 0 is undefined.

- **ID** :FR6
  **TYPE** :Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :System should display an error message when user enter value of x as 0 and y as a negative number.
  **RATIONALE** :0 raised to the power of a negative number is undefined.

## 3.2 Non-Functional Requirements

- **ID** :NFR1
  **TYPE** :Non-Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :The error message displayed should be appropriate and helpful for the user.
  **RATIONALE** :User should be able to know what went wrong.

- **ID** :NFR2
  **TYPE** :Non-Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :The text-based interface should be user friendly.
  **RATIONALE** :It should be easy for the user to use the system.

- **ID** :NFR3
  **TYPE** :Non-Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :The result displayed should be as accurate as possible.
  **RATIONALE** :Incorrect output should not be displayed.

- **ID** :NFR4
  **TYPE** :Non-Functional
  **OWNER** :Sanchit
  **DESCRIPTION** :Calculation time should be less than 1 second.
  **RATIONALE** :Waiting a long time for the output might not be desirable for the user.

## 3.3 Difficulty and Prioritization

Table 2: Difficulty and Prioritization

| Requirement ID | Priority | Difficulty |
|---|---|---|
| FR1 | High | Easy |
| FR2 | High | Easy |
| FR3 | Normal | Easy |
| FR4 | Normal | Normal |
| FR5 | High | Easy |
| FR6 | High | Normal |
| NFR1 | Low | Easy |
| NFR2 | Low | Normal |
| NFR3 | Normal | Difficult |
| NFR4 | Low | Normal |

# 4 PSEUDOCODE

## 4.1 Solution 1

**Algorithm 1:** Power-Function (x,y (input set))

begin:
1. verifyInputIsRealNumber(x,y)
2. IF x is 0 AND y $\leq$ 0 THEN
3. RAISE EXCEPTION
4. ELSE
5. Set RESULT to 1
6. Set COUNTER to 0
7. WHILE COUNTER $\neq$ y
8. UPDATE RESULT TO RESULT * x
9. INCREMENT COUNTER by 1
10. PRINT RESULT
11. READ new value for x and y
12. REPEAT the algorithm for new value

end

**Algorithm 2:** verifyInputIsRealNumber(x,y)

begin:
1. IF x,y $\in -\infty$ to $+\infty$
2. CONTINUE PROCESSING
3. else
4. RAISE EXCEPTION

end

## 4.2 Solution 2

**Algorithm 1:** Power-Function (x,y (input set))

begin:
1. verifyInputIsRealNumber(x,y)
2. verifySpecialCases(x,y)
3. Set RESULT to 1
4. Set COUNTER to 0
5. WHILE COUNTER $\neq$ y
6. UPDATE RESULT TO RESULT * x
7. INCREMENT COUNTER by 1
8. PRINT RESULT
9. READ new value for x and y
10. REPEAT the algorithm for new value

end

**Algorithm 2:** verifySpecialCases(x,y)

begin:
1. IF x is 0 AND y is 0
2. RAISE EXCEPTION
3. ELSE IF x is 0 AND y $<$ 0
4. RAISE EXCEPTION
5. ELSE
6. CONTINUE PROCESSING

end

**Algorithm 3:** verifyInputIsRealNumber(x,y)

begin:
1. IF x,y $\in -\infty$ to $+\infty$
2. CONTINUE PROCESSING
3. else
4. RAISE EXCEPTION

end

1. For Input : use READ
2. For output : use PRINT
3. For calculation : use COMPUTE
4. For Initialize: use SET

5. For Add one: use INCREMENT

### 4.3 Solution Selection

**Solution Selected for the Project is Solution 1.**

**Technical Reasons** for selection of Algorithm:

1. **Modularity:** Solution 2 provides much better modularity as there are seperate methods for error checking and special case handling.
2. **Maintenance:** Solution 1 is easy to maintain because there are less dependency and message passing between modules. Also, special case handling is done neatly with IF-ELSE mechanism.
3. **Dependencies:** Solution 1 has few dependencies as compared to Solution 2.
4. **Error Checking:** Solution 2 can provide a much more robust error checking mechanism as the error checking and special case handling is done in a seperate module. But it adds the additional computational overhead as well.
5. **Error Handling Mechanism:** Solution 1 and 2 both can provide the same level of Exception Handling mechanism.
6. **Complexity:** Solution 2 has much more complexity overhead as compared to Solution 1.

## References


[1] *"Power Functions Algebraic Representation"* `http://wmueller.com/precalculus/families/1\textunderscore41.html`

[2] *"Power Functions"* `http://www.biology.arizona.edu/biomath/tutorials/Power/Powerbasics.html`