**‘N’ digit Armstrong number checker**

**Algorithm:**

*Input: Read a number with 'n' digits  
Output: Check if the number is Amstrong or not  
  
num\_of\_digits 🡨 0  
sum 🡨 0  
temp 🡨 number  
  
if   number   ==  0  return  false  
num\_of\_digits 🡨 num\_of\_digits + 1  
number = number div 10*

*number = temp*

*if  number   ==  0  return  false  
remainder 🡨 number mod 10  
sum 🡨 sum + remainder power num\_of\_digits  
number = number div 10  
  
if sum == temp    
print Amstrong number  
else  
print Not Amstrong number*

**Best Practices:**

* *Prefer using let or const for variable declarations instead of var for block-scoped variables*
* *Use functions to define individual functionalities for better modularity.*
* *Validate the input to ensure that user enters a positive integer.*
* *Add meaningful variable & function names.*
* *Use const for immutable values to avoid accidental reassignments.*
* *Properly indent and format the code for better readability.*
* *Use strict comparison (‘===’) instead of loose comparison (‘==’).*

**Functions:**

1. *`fun()`:*

*Checks whether a given number is an Armstrong number or not:*

* *Reads a positive number from the user using `window.prompt()`.*
* *Validates the input to ensure that the user enters a valid positive number. If the input is invalid, it displays an alert to prompt the user to enter a valid number.*
* *Calls the `num\_digits()` function to determine the number of digits in the given number.*
* *Calls the `check\_condition()` function to check if the number satisfies the Armstrong condition.*
* *Displays an alert message informing the user whether the given number is an Armstrong number or not.*

1. *`num\_digits(num)`:*

*Takes a positive number `num` as input and calculates the number of digits in that number.*

* *Initializes a variable `count` to 0 to keep track of the number of digits.*
* *Uses a while loop to iterate over the digits of the number:*
* *For each iteration, it increments the `nod` variable.*
* *It then updates the `num` by dividing it by 10 to discard the last digit.*
* *The loop continues until the `num` becomes 0, and the function returns the final count of digits.*

1. *`check\_condition(num\_digits, num)`:*

*Checks whether a given number `num` satisfies the Armstrong condition with the given `num\_digits’*

* *Initializes a variable `temp` to store the original value of `num`.*
* *Initializes a variable `sum` to 0 to calculate the sum of digits raised to the power of `num\_digits`.*
* *Uses a while loop to iterate over the digits of the number:*
* *For each iteration, it calculates the remainder when dividing `num` by 10.*
* *It then adds the remainder raised to the power of `num\_digits` to the `sum`.*
* *It updates `num` by dividing it by 10.*
* *After the loop, it checks if `sum` is equal to the original `num` (stored in `temp`).*
* *The function returns `true` if `sum` is equal to `temp`, indicating that the number is an Armstrong number; otherwise, it returns `false`.*

**Boundary Condition:**

1. *Valid input: Positive integers* 
   1. *Negative number, non-numeric characters, special characters are invalid*
2. *Single digits are considered Armstrong numbers and always satisfy.*
3. *Prompt takes the input & returns it as a string or null value. Ensure to typecast it.*
4. *Boundary condition associated with the num\_digits() function.*
   1. *Increment count until number/10 > 0*
5. *Boundary condition associated with check\_condition() function.*
   1. *Iterate until the number/10 > 0*
   2. *Sum of powers of ‘n’ digits raised to nth power should be equal to the number.*
6. *Define the maximum number until which the search should be performed. The upper bound depends on the size of the Armstrong numbers you want to find.*
7. *This is not only restricted to cubic but to n digit numbers aswell.*

**Logic:**

* *Read a positive integer as a prompt in JavaScript.*
* *HTML onClick attribute calls the function fun() to prompt the input from script file.*
* *Typecast the string input into number using parseInt() or Number() functions.*
* *Calculate the number of digits in the number using a loop – divide the number by 10 in each iteration until the number becomes 0 and store the number of iterations in nod variable.*
* *Have a copy of the original variable in a temporary variable.*
* *Calculate the sum of digits each raised to the power of the individual digits obtained as remainder as a result of modulus operation and store it in a sum variable.*
* *Use loop to extract the digits:*
  + *Calculate remainder when we take number % 10*
  + *Add the remainder raised to the power of the number of digits to the sum.*
  + *Update the number by dividing by 10*
* *Compare the sum obtained with the original number.*
* *If yes, then it is a Armstrong number else it is not.*
* *Modulus operation by 10 returns the remainder that is used to extract the individual digits from the end for summating them in future.*
* *Dividing the number by 10 returns all the number except the last digit, can be used to determine the variable length instead of using (.length function)*