Documentation Report

# Task 1: Remove Punctuation

Objective:  
The purpose of this task is to remove punctuation and special characters from a given text string using a class-based implementation in Python.

## Code Explanation:

1. class remove\_punctuation: → Defines the class.  
2. \_\_init\_\_(self, text): → Constructor that initializes the input string and a list of special characters.  
3. self.spl\_char: → Stores all special characters to be removed.  
4. self.text: → Stores the provided input string.  
5. return self.checking(): → Calls the method to remove punctuation (non-standard use of return in \_\_init\_\_).  
6. checking(self): → Iterates over each special character and replaces it with a space.  
7. Prints original and cleaned string.  
8. obj1=remove\_punctuation('hello:mehwish\_sadiq') → Creates object and executes the logic.

## Sample Output:

Original: hello:mehwish\_sadiq  
Cleaned: hello mehwish sadiq

# Task 2: Sort String (Characters and Words)

Objective:  
This task sorts characters in a text string in ascending order and sorts words alphabetically using the Bubble Sort algorithm.

## Code Explanation:

1. text = 'I am Mehwish' → Input string.  
2. class Sort\_string: → Defines the class for sorting.  
3. \_\_init\_\_(self): → Splits the text into list of characters and words.  
4. sort\_char(self): → Applies Bubble Sort on characters using ASCII values.  
5. sort\_words(self): → Applies Bubble Sort on words (case-insensitive).  
6. Both sorted results are returned and displayed.  
7. Example: Characters rearranged, words alphabetically ordered.

## Sample Output:

Original: I am Mehwish  
Sorted by Characters: IMaehhimsw  
Sorted by Words: I Mehwish am

# Task 3: Luhn Algorithm (Credit Card Validation)

Objective:  
This task validates credit card numbers using the Luhn algorithm, a checksum formula used to detect errors.

## Code Explanation:

1. class lumh\_algorithm: → Defines class for Luhn algorithm.  
2. \_\_init\_\_(self, credit\_card): → Constructor takes credit card number as input.  
3. x = self.credit\_card.pop(-1): → Removes the last digit (check digit).  
4. self.credit\_card.reverse(): → Reverses the number sequence.  
5. Loop 1: Doubles digits at even positions.  
6. Loop 2: Adjusts digits greater than x by subtracting x.  
7. Append check digit back.  
8. Sum all digits and check if divisible by 10.  
9. If divisible → Valid, else → Invalid.  
10. Objects created to test with different card numbers.

## Sample Output:

valid credit card number  
unvalid credit card number  
unvalid credit card number