**IMPLEMENTATION – EX 7 :**

**Question 1:**

class TextEditor:

def \_\_init\_\_(self):

# Initialize an empty text when an instance of TextEditor is created.

self.text = ""

def load\_text(self, text):

# Load the provided text into the text editor.

self.text = text

def get\_statistics(self):

# Calculate and return statistics about the text.

char\_count = len(self.text) # Count characters

word\_count = len(self.text.split()) # Count words

sentence\_count = self.text.count('.') + self.text.count('!') + self.text.count('?') # Count sentences

return char\_count, word\_count, sentence\_count

def count\_word\_frequencies(self, top\_n):

# Count and return the top N most frequent words in the text.

words = self.text.split()

word\_freq = {}

for word in words:

word = word.strip('.,!?()[]{}":;') # Remove punctuation

word = word.lower() # Convert to lowercase

if word:

word\_freq[word] = word\_freq.get(word, 0) + 1

sorted\_word\_freq = sorted(word\_freq.items(), key=lambda x: x[1], reverse=True) # Sort by frequency

return sorted\_word\_freq[:top\_n]

def append\_text(self, text\_to\_append):

# Append the provided text to the end of the current text.

self.text += text\_to\_append

def insert\_text(self, position, text\_to\_insert):

# Insert the provided text at the specified position in the text.

self.text = self.text[:position] + text\_to\_insert + self.text[position:]

def search\_and\_replace(self, search\_text, replace\_text):

# Search for a specific text and replace it with another text in the entire document.

self.text = self.text.replace(search\_text, replace\_text)

def delete\_text(self, start, end):

# Delete a portion of the text, specified by the start and end positions.

self.text = self.text[:start] + self.text[end:]

def categorize\_text(self):

# Categorize the text based on a specific logic, but this part is left as an exercise.

# It's recommended to use external libraries for accurate categorization if needed.

categorized\_text = {} # Store categorized text and their counts

return categorized\_text

# Example usage:

editor = TextEditor()

editor.load\_text("Vicky waited for the train. The train was late. Mary and Samantha took the bus.")

char\_count, word\_count, sentence\_count = editor.get\_statistics()

print(f"Character Count: {char\_count}")

print(f"Word Count: {word\_count}")

print(f"Sentence Count: {sentence\_count}")

top\_words = editor.count\_word\_frequencies(3)

print(f"Top Words: {top\_words}")

editor.append\_text(" Appended Text.")

print("After Append:", editor.text)

editor.insert\_text(10, "Inserted")

print("After Insert:", editor.text)

editor.search\_and\_replace("sample", "modified")

print("After Replace:", editor.text)

editor.delete\_text(5, 14)

print("After Delete:", editor.text)

**OUTPUT:**

**Character Count: 80**

**Word Count: 15**

**Sentence Count: 3**

**Top Words: [('the', 3), ('train', 2), ('vicky', 1)]**

**After Append: Vicky waited for the train. The train was late. Mary and Samantha took the bus. Appended Text.**

**After Insert: Vicky waiInsertedted for the train. The train was late. Mary and Samantha took the bus. Appended Text.**

**After Replace: Vicky waiInsertedted for the train. The train was late. Mary and Samantha took the bus. Appended Text.**

**After Delete: Vickrtedted for the train. The train was late. Mary and Samantha took the bus. Appended Text.**

**Question 2:**

import re

from collections import Counter

class TextEditor:

def \_\_init\_\_(self):

self.text = ""

def load\_text(self, input\_text):

self.text = input\_text

def get\_basic\_stats(self):

char\_count = len(self.text)

word\_count = len(self.text.split())

sentence\_count = len(re.split(r'[.!?]', self.text))

return char\_count, word\_count, sentence\_count

def count\_word\_frequencies(self, top\_n):

words = re.findall(r'\w+', self.text.lower())

word\_freq = Counter(words)

return word\_freq.most\_common(top\_n)

def append\_text(self, new\_text):

self.text += new\_text

def insert\_text(self, position, new\_text):

self.text = self.text[:position] + new\_text + self.text[position:]

def search\_and\_replace(self, search\_text, replace\_text):

self.text = self.text.replace(search\_text, replace\_text)

def delete\_text(self, start, end):

self.text = self.text[:start] + self.text[end:]

def categorize\_text(self):

categories = {

"numbers": len(re.findall(r'\d+', self.text)),

"alphabets": len(re.findall(r'[a-zA-Z]+', self.text)),

"urls": len(re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\\(,]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', self.text)),

"links": len(re.findall(r'www\.[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}', self.text)),

}

categories["others"] = len(self.text.split()) - sum(categories.values())

return categories

if \_\_name\_\_ == "\_\_main\_\_":

editor = TextEditor()

input\_text = """Unfortunately, the Department hasn't bothered to keep

any of the old links,or provide cross-links into the new database-driven

website in www.apple.com"""

editor.load\_text(input\_text)

char\_count, word\_count, sentence\_count = editor.get\_basic\_stats()

print(f"Character Count: {char\_count}")

print(f"Word Count: {word\_count}")

print(f"Sentence Count: {sentence\_count}")

top\_words = editor.count\_word\_frequencies(5)

print("Top 5 words and their frequencies:")

for word, freq in top\_words:

print(f"{word}: {freq}")

categories = editor.categorize\_text()

print("Categorized Text:")

for category, count in categories.items():

print(f"{category.capitalize()}: {count}")

# Ensure total word count matches categorized count

total\_word\_count = sum(categories.values())

print(f"Total Word Count: {total\_word\_count}")

**OUTPUT:**

**Character Count: 156**

**Word Count: 22**

**Sentence Count: 3**

**Top 5 words and their frequencies:**

**the: 3**

**links: 2**

**unfortunately: 1**

**department: 1**

**hasn: 1**

**Categorized Text:**

**Numbers: 0**

**Alphabets: 27**

**Urls: 0**

**Links: 1**

**Others: -6**

**Total Word Count: 22: 5**