1. **Code Optmization:**

```
def sum_of_sqaure(N):
    result = 0
    for i in range(1,N+1):
        result += i*i
    return result
def sum_of_sqaure_optimized_version(N):
    return N*(N+1)*(2*N+1)//6
N = int(input("Please enter the no: "))
print("Unoptimized Output is:",sum_of_sqaure(N))
print("Optimized
  way:",sum_of_sqaure_optimized_version(N))
```

2. **Code Generation:**

```
def compile(expression):
    # Ensure the expression is properly
formatted as a string
    return ''.join(expression.split())

# Example usage
expression = "3 + 4 * 2"
generated_code = compile(expression)
print("Generated Python code:", generated_code)

# Evaluate the generated code
result = eval(generated_code)
```

```
print("Result of expression:", result)
```

3. Try to program to count number of tabs, lines, etc, etc, from given file.

```
def count_text_stats(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()
        num_lines = len(lines)
        num_tabs = sum(line.count('\t') for
line in lines)
        num_spaces = sum(line.count(' ') for
line in lines)
        words = ' '.join(lines).split()
        num_words = len(words)
        num_chars = sum(len(word) for word in
words)

    return num_lines, num_tabs, num_spaces,
num_words, num_chars

# Example usage
filename =
r'C:\Users\RONIT\Desktop\cd\sample.txt'# Change
this to your file path
```

```
line_count, tab_count, space_count, word_count,
char_count = count_text_stats(filename)
print("Number of lines:", line_count)
print("Number of tabs:", tab_count)
print("Number of spaces:", space_count)
print("Number of words:", word_count)
print("Number of characters:", char_count)
```

5)

Write LEX specifications and necessary C code that reads English words from a text file and calculates the count of words that starts with a vowel. The program appends the current value of the counter to every occurrences of such word. The program should also compute total numbers of words read.

→

```
import re


def append_counter(word, count):

    return f"{word}{count}"


def process_file(filename):

    with open(filename, 'r') as file:

        content = file.read()
    vowel_count = 0
    total_words = 0

    def replace_word(match):
        nonlocal vowel_count, total_words
        word = match.group(0)
        if re.match(r'^[aeiouAEIOU]', word):
            vowel_count += 1
            word = append_counter(word,
vowel_count)
        total_words += 1
        return word

    processed_content = re.sub(r'\b[a-zA-Z]+\b',
replace_word, content)

    print(processed_content)
    print(f"\nTotal words: {total_words}")


if __name__ == "__main__":
    filename = 'C:\\Users\\RONIT\\Desktop\\cd\\sample.txt'
    process_file(filename)
```

5)

Write LEX specifications and necessary C code that reads English words from a text file and replaces every occurrence of the sub string ‘abc’ with ‘ABC’. The program should also compute number of characters, words and lines read. It should not consider and count any line(s) that begin with a symbol “#”.

→

```
import re


def count_chars_and_words(text):
    char_count = len(text)
```

```python
        word_count = len(re.findall(r'\b\w+\b', text))
    return char_count, word_count


def process_file(filename):
    char_count = 0
    word_count = 0
    line_count = 0

    with open(filename, 'r') as file:
        for line in file:
            if line.startswith('#'):
                continue
            line_count += 1
            line = re.sub(r'abc', 'ABC', line)
            print(line, end='')
            chars, words = count_chars_and_words(line)
            char_count += chars
            word_count += words

    print(f"\nCharacters: {char_count}")
    print(f"Words: {word_count}")
    print(f"Lines: {line_count}")

if __name__ == "__main__":
                                 filename       =
'C:\\Users\\RONIT\\Desktop\\cd\\sample.txt'       #
Define the file path here
    process_file(filename)
```

3.Write a lex program to count number of characters, words and lines in a given input text file. Create an output text file that consists of the content of the input file as well as line numbers

```
def count_chars_and_words(text):
    char_count = len(text)
    word_count = len(text.split())
```

```python
    return char_count, word_count


def process_file(input_filename, output_filename):
    char_count = 0
    word_count = 0
    line_count = 0


    with open(input_filename, 'r') as input_file, open(output_filename, 'w') as output_file:
        for line in input_file:
            line_count += 1
            char_count += len(line)
            word_count += len(line.split())


                # Write line number and content to output file
                output_file.write(f"{line_count}: {line}")
```

2.

```
    return char_count, word_count, line_count


if __name__ == "__main__":
    input_filename = 'input.txt'    # Provide the input file path
    output_filename = 'output.txt' # Provide the output file path


    char_count, word_count, line_count = process_file(input_filename, output_filename)


    print(f"Characters: {char_count}")
    print(f"Words: {word_count}")
    print(f"Lines: {line_count}")
```

2.

Implement the Lexical analyzer for the given language. The lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value.

```
import re
```

```
def lexical_analyzer(input_text):
    tokens = []
    current_token = ''
    in_comment = False

    # Regular expressions for tokens
    identifier_regex = r'[a-zA-Z][a-zA-Z0-9]*'
    number_regex = r'\d+'
    symbol_regex = r'[\[\]()+\-*/=]'

    for line in input_text.split('\n'):
        # Remove leading and trailing whitespace
        line = line.strip()
        # Skip empty lines
        if not line:
            continue

        # Remove comments
        line = re.sub(r'#.*', '', line)

        # Split line into tokens
        for token in re.findall(f'{identifier_regex}|{number_regex}|{symbol_regex}', line):
            tokens.append(token)

    return tokens

if __name__ == "__main__":
    # Example input text
    input_text = """
    # This is a comment
    x = 10
    y = 20
    result = x + y
    """


    tokens = lexical_analyzer(input_text)
    print("Tokens:", tokens)
```

GITHUB LEX COMPILOR:
https://github.com/Tempo4u/system272.git