#3. Design your own tokenizer (30% for BS students – 20% for MS students)

```
import re
from collections import Counter


def tokenizer(text):
    # Regular expression to tokenize text
    token_pattern = r"\b\w+\b"
    tokens = re.findall(token_pattern, text)
    return tokens

# Load the corpus from "elon_tesla.txt"
with open("elo_tesla.txt", "r", encoding="utf-8") as file:
    corpus = file.read()

# Tokenize the corpus
tokens = tokenizer(corpus)

# a) Total number of tokens
total_tokens = len(tokens)

# b) Number of unique tokens
unique_tokens = len(set(tokens))

# c) Top 10 most common tokens (case-sensitive)
common_tokens = Counter(tokens).most_common(10)

# d) Top 10 most common tokens (case-insensitive)
tokens_lowercase = [token.lower() for token in tokens]
common_tokens_lowercase = Counter(tokens_lowercase).most_common(10)

# e) Function to find tokens with more than 10 letters
def long_tokens(tokens, min_length=10):
    long_tokens_list = [token for token in tokens if len(token) > min_length]
    return Counter(long_tokens_list).most_common(10)

long_tokens_result = long_tokens(tokens)

# f) Longest token in the corpus
longest_token = max(tokens, key=len)

# Print the results
print(f"Total tokens: {total_tokens}")
print(f"Unique tokens: {unique_tokens}")
print(f"Top 10 most common tokens (case-sensitive): {common_tokens}")
print(f"Top 10 most common tokens (case-insensitive): {common_tokens_lowercase}")
print(f"Tokens with more than 10 letters: {long_tokens_result}")
print(f"Longest token: {longest_token}")
```