U N I V E R S I T Y   O F   K A R A C H I
Department of Computer Science

U B I T

COMPILER CONSTRUCTION

GHUFRAN AHMED
EB21103039

SYED FURQUAN AKHTAR
EB21103127

```
# Code Lexical analyzer on Word Splitter
# Author: GHUFRAN AHMED
# Seat No: EB21103039

import re


def tokenize(code, patterns):
    """
    Function to tokenize the code into different types of tokens.
    """
    tokens = []
    for pattern, token_type in patterns:
        matches = re.finditer(pattern, code)
        for match in matches:
            tokens.append((match.group(), token_type))
    return tokens


def remove_comments(code):
    """
    Function to remove comments from the code.
    """
    # Remove single-line comments
    code = re.sub('//.*', '', code)
    # Remove multi-line comments
    code = re.sub('/\*.*?\*/', '', code, flags=re.DOTALL)
    return code


def extract_tokens(code):
    """
    Function to extract tokens from the code.
    """
    code = remove_comments(code)

    # Predefined token patterns for a specific language
    patterns = [
        (r'\b(?:if|else|while|for)\b', 'KEYWORD'),  # Keywords
        (r'\b[a-zA-Z_]\w*\b', 'IDENTIFIER'),  # Identifiers
        (r'\b\d+\b', 'LITERAL'),  # Numeric literals
        (r'[+\-*/=<>]', 'OPERATOR'),  # Operators
        (r'[;(),.]', 'SEPARATOR')  # Separators
    ]

    tokens = tokenize(code, patterns)
    return tokens


def display_words(token_list):
    """
    Function to display the extracted words.
    """
    print("Extracted Words:")
    words = [token for token, token_type in token_list if token_type == 'IDENTIFIER']
    print(words)
```

```
def display_token_types(token_list):
    """
    Function to display the token type with each word.
    """
    print("Token Types:")
    for token, token_type in token_list:
        print(f"{token}: {token_type}")


def display_token_counts(token_list):
    """
    Function to display the token counts.
    """
    # Count token types
    token_counts = {}
    for _, token_type in token_list:
        if token_type in token_counts:
            token_counts[token_type] += 1
        else:
            token_counts[token_type] = 1

    # Print the counts
    print("\nToken Counts:")
    for token_type, count in token_counts.items():
        print(f"{token_type} count: {count}")


def main():
    """
    Main function to orchestrate the program flow.
    """
    code_option = input("Choose an option:\n1. Type the code\n2. Read from a file\n")
    code = ""

    if code_option == "1":
        while True:
            line = input("Enter code line (leave empty to finish): ")
            if not line:
                break
            code += line + "\n"
    elif code_option == "2":
        filename = input("Enter the file name: ")
        try:
            with open(filename, "r") as file:
                code = file.read()
        except FileNotFoundError:
            print("File not found.")
            return
    else:
        print("Invalid option.")
        return

    extracted_tokens = extract_tokens(code)
    # Store the tokens in a list or any desired data structure
    token_list = extracted_tokens

    display_words_prompt = input("Do you want to display only the extracted words? (y/n):
").lower()

    if display_words_prompt == 'y':
        display_words(token_list)

    display_token_types_prompt = input("Do you want to display the token types with each
word? (y/n): ").lower()

    if display_token_types_prompt == 'y':
        display_token_types(token_list)

    display_token_counts_prompt = input("Do you want to display the token counts? (y/n):
").lower()

    if display_token_counts_prompt == 'y':
        display_token_counts(token_list)

    # Display the author name
    print("\nAuthor: GHUFRAN AHMED")


if __name__ == "__main__":
    main()
```