```
import string

INT_LIT = 'INT_LIT'
IDENT = 'IDENT'
ADD_OP = 'ADD_OP'
SUB_OP = 'SUB_OP'
MULT_OP = 'MULT_OP'
DIV_OP = 'DIV_OP'
LEFT_PAREN = 'LEFT_PAREN'
RIGHT_PAREN = 'RIGHT_PAREN'
EOF = 'EOF'
UNKNOWN = 'UNKNOWN'

lookup_table = {
   '+': ADD_OP,
   '-': SUB_OP,
   '*': MULT_OP,
   '/': DIV_OP,
   '(': LEFT_PAREN,
   ')': RIGHT_PAREN
}

def is_letter(char):
   return char in string.ascii_letters

def is_digit(char):
```

```
    return char in string.digits


def lexer(input_string):
    index = 0
    tokens = []

    while index < len(input_string):
        current_char = input_string[index]

        if current_char.isspace():
            index += 1
            continue

        elif is_letter(current_char):
            start = index
            while index < len(input_string) and (is_letter(input_string[index]) or
is_digit(input_string[index])):
                index += 1
            lexeme = input_string[start:index]
            tokens.append((IDENT, lexeme))

        elif is_digit(current_char):
            start = index
            while index < len(input_string) and is_digit(input_string[index]):
                index += 1
            lexeme = input_string[start:index]
```

```
            tokens.append((INT_LIT, lexeme))

        elif current_char in lookup_table:
            tokens.append((lookup_table[current_char], current_char))
            index += 1

        else:
            tokens.append((UNKNOWN, current_char))
            index += 1

    tokens.append((EOF, 'EOF'))
    return tokens

expression = "sum + 25 * (value - 7)"
tokens = lexer(expression)

for token_type, lexeme in tokens:
    print(f"Next token is: {token_type}, Next lexeme is: {lexeme}")
```