Ex. No : 4 **Token Separation using high level language**

Date :

AIM:

To write a python program for token separation using high level language.

Procedure:

- Split the text into lines using the splitlines() method.
- Iterate over the lines of text using the for loop.
- For each line, tokenize the line using the tokenize.tokenize() function.
- Append each token to the tokens list.
- Return the tokens list.

Code:

This code will tokenize the text by splitting it at each space, punctuation mark, and newline character. It will also return empty strings for any whitespace characters between tokens.

```
import tokenize
import io

def tokenize_text(input_text):
    """Tokenizes the given text and returns a list of tokens.

    Args:
        input_text: The text to be tokenized.

    Returns:
        A list of tokens.
    """

    tokens = []
    for tok in tokenize.tokenize(io.BytesIO(input_text.encode('utf-8')).readline):
        tokens.append(tok.string)
    return tokens

# Example usage:
text = "Virat Kohli, a cricket maestro, exemplifies unparalleled skill, tenacity, and charisma on the field, captivating fans worldwide."

tokens = tokenize_text(text)

print(tokens)
```

Output:

Result:

Thus, we executed a python program for token separation using high level language.

Ex. No : 6 a) **Count Of Positive And Negative Numbers**

Date :

AIM:

To write a Lex program for counting the positive and negative numbers.

Procedure:

- This program will recognize any string that consists of alphanumeric characters, underscores, periods, percent signs, plus signs, and hyphens, followed by an @ symbol, followed by another string that consists of alphanumeric characters, periods, and hyphens, followed by a two-letter domain extension
- If the string matches this pattern, the program will print "Valid email address."
- Otherwise, it will print "Invalid email address.

Code:

```
%%

[-+]?[0-9]* {
  if (yytext[0] == '-') {
    negative_numbers++;
  } else {
    positive_numbers++;
  }
}

%%

int main() {
  int positive_numbers = 0;
  int negative_numbers = 0;

  yylex();

  printf("Positive numbers: %d\n", positive_numbers);
  printf("Negative numbers: %d\n", negative_numbers);

  return 0;
}
```

Output:

```
$ lex count numbers.l
$ gcc lex.yy.c -o count numbers
$ ./count numbers
Enter a stream of numbers: 1, 2, -8.4, -15.6, 76, 34
Positive numbers: 4
Negative numbers: 2
```

Result:

Thus, we executed a Lex program for counting the positive and negative numbers.

Ex. No: 6 b) **Count of Number of Words, Characters and Lines**
Date :

AIM:

To write a Lex program for counting number of words, characters and lines.

Procedure:

- Program will recognize any sequence of alphanumeric characters, underscores, apostrophes, and hyphens asa word. It will also recognize spaces, tabs, and newlines as whitespace.
- When it encounters whitespace, it will increment the character count and the line count if it is a newline.
- It will also increment the word count if it just encountered a word.

Code:

```
%%

[a-zA-Z0-9_'-]+ {
  word_count++;
}

[[ \t\n]] {
  character_count++;
  if (yytext[0] == '\n') {
    line_count++;
  }
}

. {
  character_count++;
}
%%

int main() {
  int word_count = 0;
  int character_count = 0;
  int line_count = 0;

  yylex();

  printf("Word count: %d\n", word_count);
  printf("Character count: %d\n", character_count);
  printf("Line count: %d\n", line_count);

  return 0;
}
```

Output:

```
$ lex count_words_characters_lines.l
$ gcc lex.yy.c -o count_words_characters_lines
$ ./count_words_characters_lines
Enter a stream of text:
Hello there how are you.
Word count: 5
Character count: 23
Line count: 1
```

Result:

Thus, we executed a Lex program for counting number of words, characters and lines.

Ex. No : 6 c) **Count the Vowels and Consonants**

Date :

Aim:

To write a Lex program for counting vowels and consonants.

Procedure:

- This program will recognize any vowel (a, e, i, o, u) as a vowel, and any other letter as a consonant.
- When it encounters a vowel or consonant, it will increment/decrement the appropriate counter.

Code:

```
%%

[aeiou] {
  vowel_count++;
}

[bcdfghjklmnpqrstvwxyz] {
  consonant_count++;
}

%%

int main() {
  int vowel_count = 0;
  int consonant_count = 0;

  yylex();

  printf("Vowel count: %d\n", vowel_count);
  printf("Consonant count: %d\n", consonant_count);

  return 0;
}
```

Output:

```
$ lex count_vowels_consonants.l
$ gcc lex.yy.c -o count_vowels_consonants
$ ./count_vowels_consonants
Enter a stream of text:
Hello how are you.

Vowel count: 7
Consonant count: 7
```

Result:

Thus, we executed a Lex program for counting vowels and consonants.