University of Asia Pacific

CSE 430

Compiler Design Lab

Assignment – 1

Write a Program for Lexical Analysis

Name: Md. Azim Islam

Registration: 19201026

Section: A2

Course Code: CSE 430

Semester: Spring 2023

Baivab Das

Lecturer, University of Asia Pacific, Dhaka

CODE

```
import re
from sys import stdin
from collections import defaultdict

kw = {'auto', 'break', 'case', 'char', 'const', 'continue',
      'default', 'do', 'double', 'else', 'enum', 'extern', 'float',
      'for', 'goto', 'if', 'int', 'long', 'register', 'return', 'short',
      'signed', 'sizeof', 'static', 'struct',
      'switch', 'typedef', 'union', 'unsigned', 'void', 'volatile',
'while'}


d = defaultdict(set)

def tokenize(word, d):
    if word in kw:
        d['Keyword'].add(word)
        return
    #Function Identifier
    p =
r"(?P<func>([_]|[a-z]|[A-Z])(([_]|[a-z]|[A-Z]|[0-9])*))(?:\(.*\))"
    m = re.match(p, word)
    if m:
        d['Functions'].add(m.group('func'))

    #Atomic Variable Declaration Indentifier
    #a;|a,
    p =
r"(?P<variable>([_]|[a-z]|[A-Z])(([_]|[a-z]|[A-Z]|[0-9])*))(?P<punc>[,|;
])?"
    m = re.match(p, word)
    if m:
        d['Identifiers'].add(m.group('variable'))
    #Atomic Variable Assignment Identifier
```

```
    #a=10;
    p =
r"(?P<variable>([_]|[a-z]|[A-Z])(([_]|[a-z]|[A-Z]|[0-9])*))=(?P<constant
>[0-9]+)(?P<punc>[,|;])?"
    m = re.match(p, word)
    if m:
        d['Identifiers'].add(m.group('variable'))
        d['Constants'].add(m.group('constant'))


    p = r"[*]|[+]|[/]|[-]|[=]"
    m = re.findall(p, word)
    for i in m:
        d['Arithmetic Operator'].add(i)


    p = r"[,]|[;]|[:]|[!]"
    m = re.findall(p, word)
    for i in m:
        d['Punctuations'].add(i)


    #invalid at -10m
    p =
r"(?<!([_]|[A-Z]|[a-z]))([=][*]|[+]|[/]|[-])?(?P<const>([0-9])+)([=][*]|
[+]|[/]|[-])?(?!([_]|[A-Z]|[a-z]))(?P<punc>[,|;])?"
    m = re.match(p, word)
    if m:
        d['Constants'].add(m.group('const'))


    p = r"[{]|[}]|[\[]]|[\]]|[\(]|[\)]"
    m = re.findall(p, word)
    for i in m:
        d['Parenthesis'].add(i)
```

```
for line in stdin:
    if not line.strip():
        break

    if ord(line[0]) == 47:
        continue

    for word in line.split():
        tokenize(word, d)

for key in d:
    print(f"{key} [{len(d[key])}]: {' '.join(d[key])}")
```

INPUT

```
void main()
{
int a, b, c;
//comment
int a = b*c + 10;
}
```

OUTPUT

```
Keyword [2]: void int
Functions [1]: main
Identifiers [4]: a c b main
Parenthesis [4]: ) } ( {
Punctuations [2]: , ;
Arithmetic Operator [3]: * = +
Constants [1]: 10
```