import re

f = open('exp1.c','r')

op = { '=': 'Assignment Operator','+': 'Addition Operator or sum', '-'

: 'Subtraction Operator or difference', '/' : 'Division Operator or quotient ', '\*': 'Multiplication Operator or product', '++' : 'increment Operator or increase by 1', '--' : 'Decrement Operator or decrease by one'}

op\_key = op.keys()

com = {r'//' : 'for Single Line Comment ',r'/\*' : 'for Multiline Comment Start', r'\*/' : 'for Multiline Comment End', '/\*\*/' : ' for Empty Multiline comment'}

com\_key = com.keys()

hdr = {'.h':'header file'} hdr\_key = hdr.keys()

standard\_hdr\_files = {'<stdio.h>':'Standard Input Output Header','<string.h>':'String Manipulation Library','<math.h>':'mathematics function library'}

macros = {r'#\w+' : 'macro'} macros\_keys = macros.keys()

dt = {'int': 'Integer datatype ', 'char': 'Character datatype','float'

: 'Floating Point datatype','long': 'long int datatype'} dt\_keys = dt.keys()

kd = {'return' : 'keyword that returns a value from a block'} kd\_keys = kd.keys()

delimiter = {';':'semicolon (;)'} delimiter\_keys = delimiter.keys()

block = {'{' : 'Statement block Body Open', '}':'Statement block Body Closed'}

block\_keys = block.keys()

inbuilt = {'printf':'printf prints its argument on the console','pow':'takes power of a number'}

non\_identifiers = ['\_','- ','+','/','\*','`','~','!','@','#','$','%','^','&','\*','(',')','=','|', '"',':',';','{'

,'}','[',']','<','>','?','/']

numbers = ['0','1','2','3','4','5','6','7','8','9','10']

# Flags

dtFlag = False i = f.read()

count = 0

prog = i.split('\n') for l in prog:

tokens=line.split(" ") for token in tokens:

if token in block\_keys: print(block[token])

if token in op\_keys: print("operator is ",op[token])

if token in com\_keys:

print("comments types are ",com[token]) if token in macros\_keys:

print("macro's are ",macros[token]) if '.h' in token:

print("headers are ",standard\_hdr\_files[token]) if '()' in token:

print("function named",token)

if dtFlag==True and (token not in non\_identifiers) and ('()' not in token):

print("identifier is ",token) if token in dt\_keys:

print("type is ",dt[token]) dataFlag==True

if token in kd\_keys: print("keyword is ",kd[token])

if token in delimiter\_keys: print("delimitter is ",delimiter[token])

if token in numbers:

print("numerals are ",token,type(int(token))) if token in builtin\_functions:

print (built\_in functions [token]) dataFlag==False

f.close()