# COMPILER DESIGN AS 5

import re

data\_type={'int','float','char','string'}

regex = {

    'Keyword': re.compile(r'\b(?:if|else|while|return|int|float|char|string|main|void)\b'),

    'Identifier': re.compile(r'\b[a-zA-Z\_][a-zA-Z0-9\_]\*\b'),

    'Float': re.compile(r'\b\d+\.\d+\b'),

    'Int': re.compile(r'\b\d+\b'),

    'String': re.compile(r'"[^"]\*"'),

    'Char': re.compile(r"'[^']'"),

    'Operator': re.compile(r'[+\-\*/%==!=<>]=?|&&|\|\|'),

    'Delimiter': re.compile(r'[;,{}()]')

}

symbol\_table, error\_log = {}, []

def insert(var, type, lno):

    if var not in symbol\_table:

        symbol\_table[var] = {

            'type': type,

            'memory\_location': hex(hash(id(var))),

            'line': lno }

def lexer():

    n = int(input("Enter your source code length: "))

    tokens, in\_comment = [], False

    combined = ""

    for i in range(1,n+1):

        line = input()

        line = re.sub(r'//.\*$', '', line)

        combined+=line

        if '/\*' in line:

            in\_comment = True

        if '\*/' in line and in\_comment:

            in\_comment = False

            line = line.split('\*/', 1)[1]

        if in\_comment:

            continue

        line = line.strip()

        while line:

            matched = False

            for key, pattern in regex.items():

                match = pattern.match(line)

                if match:

                    variable = match.group(0)

                    var\_type = key

                    if ((var\_type == 'Keyword') and (variable in data\_type)):

                        next\_token = line[len(variable):].strip().split()[0] if line[len(variable):].strip() else None

                        if next\_token and regex['Identifier'].fullmatch(next\_token):

                            insert(next\_token, variable, i)

                    tokens.append((var\_type, variable))

                    line = line[len(variable):].strip()

                    matched = True

                    break

            if not matched:

                error\_log.append(f"Invalid character '{line[0]}' at line {i}")

                break

    if any(lit == "" for lit in re.findall(r'"(.\*?)"', combined)):

        error\_log.append("Unterminated string literal found.")

    if combined.count('/\*') != combined.count('\*/'):

        error\_log.append("Unterminated multi-line comment found.")

    return tokens

tokens = lexer()

print("Tokens list\n","-"\*60)

print(tokens)

print("-"\*60)

print("\nSymbol Table:")

print(f"{'Identifier':<15}{'Data Type':<10}{'Memory Address':<20}{'Line number'}")

for sym, info in symbol\_table.items():

    print(f"{sym:<15}{info['type']:<10}{info['memory\_location']:<20}{info['line']}")

print("\nErrors:")

if error\_log:

    with open("errorlog.log","w") as f:

        f.write("\n".join(error\_log))

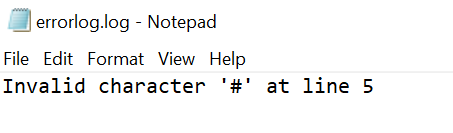
        print("Error Successfully logged in errorlog.log")

else:

    print("No errors found.")

## TEST CASE 1 :





## TEST CASE 2 :

