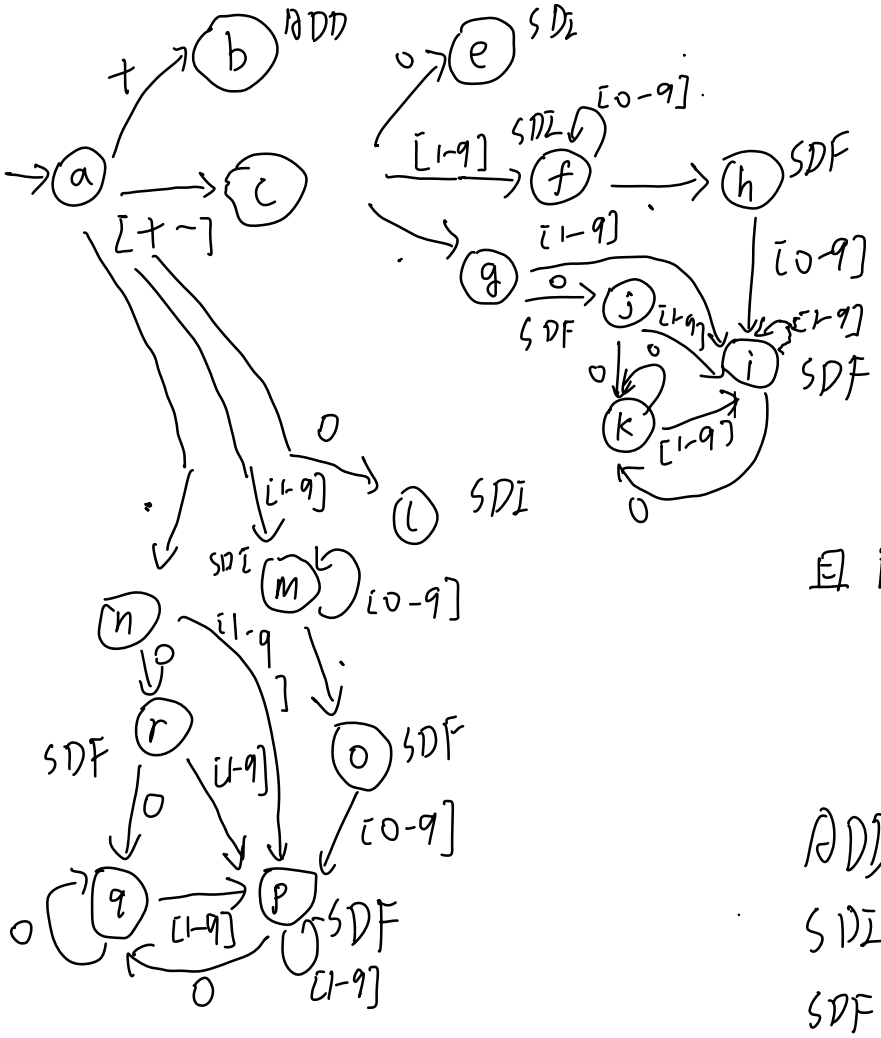
习题4.2 扩展例4.2的UDI为SDI = {+, −}UDI ∪ UDI，扩展UDF为SDF = {+, −}UDF ∪ UDF。试构建 σ-DFA({ADD, SDI, SDF})，并以 12+34.0+.56#输入串运行之，写出运行结果。 现在 SDI → [+−]? UDI, SDF → [+−]? UDF, ADD → +

由例4.2，可画出 G-DFA({ADD, SDI, SDF})



且 $F^P = \{\{b\}, \{e, f, l, m\}, \{h, i, j, o, p, r\}\}$

$ADD = \{w \in \Sigma \mid \tilde{\delta}(a, w) \in \{b\}\}$

$SDI = \{w \in \Sigma \mid \tilde{\delta}(a, w) \in \{e, f, l, m\}\}$

$SDF = \{w \in \Sigma \mid \tilde{\delta}(a, w) \in \{h, i, j, o, p, r\}\}$

$\Psi = \{(b, ADD), (e, SDI), (f, SDI), (l, SDI), (m, SDI), (h, SDF), (i, SDF), (j, SDF), (o, SDF), (p, SDF), (r, SDF)\}$

最后运行结果为 (SDI, 12), (SDI, +34.0), (SDF, +.56]

大作业第一部分

计算机越杰 2101 李昀蔚

使用 python 构造了 scanner 函数：

对于 INT，IF 这样的关键字，我们把他做成一个可查询的字典，这样当我们按照正常读 ID 的逻辑时，如果发现读出的最终结果与字典匹配，则读出结果是与之对应的关键字。

对于 LBR 这类的符号，直接匹配单字符即可。

最后单独处理了带+-号的 num。

```
KEYWORDS = {
    'int': 'INT', 'if': 'IF', 'else': 'ELSE', 'while': 'WHILE',
    'return': 'RETURN', 'void': 'VOID', 'and': 'AND', 'or': 'OR',
    'print': 'PRINT'
}

SINGLE_CHAR_TOKENS = {
    '+': 'ADD', '*': 'MUL', '(': 'LPA', ')': 'RPA',
    '{': 'LBR', '}': 'RBR', ';': 'SCO', ',': 'CMA'
}

ROP_START = {'<', '=', '>', '!'}

import re

def is_letter(ch):
    return ch.isalpha()

def is_digit(ch):
    return ch.isdigit()

def is_alnum(ch):
    return ch.isalnum()

def scanner(input_line):
    tokens = []
    i = 0
    n = len(input_line)
```

```
while i < n:
    ch = input_line[i]

    # 跳过空白
    if ch.isspace():
        i += 1
        continue

    # 标识符 ID / 关键字
    if is_letter(ch): #[a-zA-Z]
        start = i
        while i < n and is_alnum(input_line[i]): #[a-zA-Z0-9]
            i += 1
        lexeme = input_line[start:i]
        kind = KEYWORDS.get(lexeme, 'ID') #判断是否是特殊
        tokens.append((kind, lexeme))
        continue

    # 数字 NUM（可带正负号）
    if is_digit(ch) or (ch in '+-' and i + 1 < n and is_digit(input_line[i+1])):
        start = i
        if ch in '+-':
            i += 1
        while i < n and is_digit(input_line[i]):
            i += 1
        tokens.append(('NUM', input_line[start:i]))
        continue

    # 双字符运算符（ROP）
    if ch in ROP_START:
        if i + 1 < n and input_line[i + 1] == '=':
            tokens.append(('ROP', ch + '='))
            i += 2
        else:
            tokens.append(('ROP', ch))
            i += 1
        continue

    # 剩余的单字符词法单元
    if ch in SINGLE_CHAR_TOKENS:
        tokens.append((SINGLE_CHAR_TOKENS[ch], ch))
        i += 1
        continue
```

```
            # 其他非法字符
            tokens.append(('ERR', ch))
            i += 1

    return tokens


if __name__ == '__main__':
    test_line = """
    int raw(int x;){
        y = x + 5;
        return y;
    };
    void foo(int y;){
        int z;
        void bar(intx; int sooQ;){
            if(x > 3) bar(x / 3, sooO, ) else z = soo(x);
            print z;
        };
        bar(y, rawO, );
    };
    foo(6, );
    """
    print("输入代码：", test_line)
    result = scanner(test_line)
    print("词法分析输出：")
    for token in result:
        print(token)
```

可以得到结果为：

```
('INT', 'int')
('ID', 'raw')
('LPA', '(')
('INT', 'int')
('ID', 'x')
('SCO', ';')
('RPA', ')')
('LBR', '{')
('ID', 'y')
('ROP', '=')
('ID', 'x')
('ADD', '+')
```

('NUM', '5')
('SCO', ';')
('RETURN', 'return')
('ID', 'y')
('SCO', ';')
('RBR', '}')
('SCO', ';')
('VOID', 'void')
('ID', 'foo')
('LPA', '(')
('INT', 'int')
('ID', 'y')
('SCO', ';')
('RPA', ')')
('LBR', '{')
('INT', 'int')
('ID', 'z')
('SCO', ';')
('VOID', 'void')
('ID', 'bar')
('LPA', '(')
('ID', 'intx')
('SCO', ';')
('INT', 'int')
('ID', 'sooQ')
('SCO', ';')
('RPA', ')')
('LBR', '{')
('IF', 'if')
('LPA', '(')
('ID', 'x')
('ROP', '>')
('NUM', '3')
('RPA', ')')
('ID', 'bar')
('LPA', '(')
('ID', 'x')
('ERR', '/')
('NUM', '3')
('CMA', ',')
('ID', 'sooO')
('CMA', ',')
('RPA', ')')
('ELSE', 'else')

('ID', 'z')
('ROP', '=')
('ID', 'soo')
('LPA', '(')
('ID', 'x')
('RPA', ')')
('SCO', ';')
('PRINT', 'print')
('ID', 'z')
('SCO', ';')
('RBR', '}')
('SCO', ';')
('ID', 'bar')
('LPA', '(')
('ID', 'y')
('CMA', ',')
('ID', 'rawO')
('CMA', ',')
('RPA', ')')
('SCO', ';')
('RBR', '}')
('SCO', ';')
('ID', 'foo')
('LPA', '(')
('NUM', '6')
('CMA', ',')
('RPA', ')')
('SCO', ';')