Programming Language HW #1

2022001167 이조에

#0 parsing table

State	N	+	-	*	/	$	E	T
0	S3						1	2
1		S4	S5			acc		
2		R3	R3	S6	S7	R3		
3	R6	R6	R6	R6	R6	R6		
4	S3							8
5	S3							9
6	S10							
7	S11							
8		R1	R1	S6	S7	R1		
9		R2	R2	S6	S7	R2		
10	R4	R4	R4	R4	R4	R4		
11	R5	R5	R6	R6	R6	R5		

#3 recursive-descent parsing 을 위해 만든 right recursion rule

1. $E \rightarrow TE'$
2. $E' \rightarrow +TE' \mid -TE' \mid \epsilon$
3. $T \rightarrow NT'$
4. $T' \rightarrow *NT' \mid /NT' \mid \epsilon$
5. $N \rightarrow$ number

---

e.g. 5+5/5

```
#1

Lexemes: ['5', '+', '5', '/', '5', '$']
Tokens: ['N', '+', 'N', '/', 'N', '$']
```

5 는 number

'+', '/' 는 operator

'$'는 input 이 끝났음을 알림

```
#2

Tracing Start!!
+-----------------------------+-----------------------------------+
|Step|          Stack         |          Input          |    Action       |
+----+------------------------+-------------------------+-----------------+
(00) |0                       |N + N / N $              | Shift 3         |
(01) |0 N 3                   |+ N / N $                | Reduce 6        |
(02) |0 T 2                   |+ N / N $                | Reduce 3        |
(03) |0 E 1                   |+ N / N $                | Shift 4         |
(04) |0 E 1 + 4               |N / N $                  | Shift 3         |
(05) |0 E 1 + 4 N 3           |/ N $                    | Reduce 6        |
(06) |0 E 1 + 4 T 8           |/ N $                    | Shift 7         |
(07) |0 E 1 + 4 T 8 / 7       |N $                      | Shift 11        |
(08) |0 E 1 + 4 T 8 / 7 N 11  |$                        | Reduce 5        |
(09) |0 E 1 + 4 T 8           |$                        | Reduce 1        |
(10) |0 E 1                   |$                        | acc             |

Parsing success!

Result: 6.0
```

Reduce x 를 중점적으로 보면,

Step 01: N → T (6)

Step 02: T → E (3)

Step 05: N → T (6)

Step 08: T/N → T (5)

Step 09: E+T → E (1)

```
#3

LET'S START!
Enter E
Enter T
Enter T'
epsilon
Exit T'
Exit T
Enter E'
Enter T
Enter T'
Enter T'
epsilon
Exit T'
Exit T'
Exit T
Enter E'
epsilon
Exit E'
Exit E'
Exit E
Recursive-Descent parsing Result: 6.0
```

Enter 를 중점적으로 보면,

E: E → TE'

T: T → NT' | 5 인식 | T' 호출

E': E' → +TE' | + 인식

T: T →NT' | 5 인식 | T' 호출

T': T' →/NT' | / 인식, 5 인식 | T' 호출 → epsilon

T': epsilon

E': epsilon

---

e.g. 100/13*4

```
#1

Lexemes: ['100', '/', '13', '*', '4', '$']
Tokens: ['N', '/', 'N', '*', 'N', '$']
```

100, 13, 4 는 number

'/', '*'는 operator

'$'는 input 이 끝났음을 알림

```
#2

Tracing Start!!
+-----------------------------+-----------------------------------+
|Step|          Stack         |          Input          |    Action       |
+----+------------------------+-------------------------+-----------------+
(00) |0                       |N / N * N $              | Shift 3         |
(01) |0 N 3                   |/ N * N $                | Reduce 6        |
(02) |0 T 2                   |/ N * N $                | Shift 7         |
(03) |0 T 2 / 7               |N * N $                  | Shift 11        |
(04) |0 T 2 / 7 N 11          |* N $                    | Reduce 5        |
(05) |0 T 2                   |* N $                    | Shift 6         |
(06) |0 T 2 * 6               |N $                      | Shift 10        |
(07) |0 T 2 * 6 N 10          |$                        | Reduce 4        |
(08) |0 T 2                   |$                        | Reduce 3        |
(09) |0 E 1                   |$                        | acc             |

Parsing success!

Result: 30.76923076923077
```

Reduce x 를 중점적으로 보면,

Step 01: N → T (6)

Step 04: T/N → T (5)

Step 07: T*N → T (4)

Step 08: T → E (3)

```
#3

LET'S START!
Enter E
Enter T
Enter T'
Enter T'
Enter T'
epsilon
Exit T'
Exit T'
Exit T'
Exit T
Enter E'
epsilon
Exit E'
Exit E
Recursive-Descent parsing Result: 30.76923076923077
```

Enter 를 중점적으로 보면,

E: E → TE'

T: T → NT' | 100 인식 | T' 호출

T': T' → /NT' | / 인식, 13 인식 | T' 호출

T' T' → *NT' | * 인식, 4 인식 | T' 호출

T': epsilon

E': epsilon

---

e.g. 20*10+30/10-4

```
#1

Lexemes: ['20', '*', '10', '+', '30', '/', '10', '-', '4', '$']
Tokens: ['N', '*', 'N', '+', 'N', '/', 'N', '-', 'N', '$']
```

20, 10, 30, 10, 4 는 number

'*', '+', '/', '-'는 operator

'$'는 input 이 끝났음을 알림

```
#2

Tracing Start!!
+----------------------------+--------------------------------------+
|Step|          Stack        |          Input       |    Action     |
+----+-----------------------+--------------------------------------+
(00) |0                      |N * N + N / N - N $   | Shift 3       |
(01) |0 N 3                  |* N + N / N - N $     | Reduce 6      |
(02) |0 T 2                  |* N + N / N - N $     | Shift 6       |
(03) |0 T 2 * 6              |N + N / N - N $       | Shift 10      |
(04) |0 T 2 * 6 N 10         |+ N / N - N $         | Reduce 4      |
(05) |0 T 2                  |+ N / N - N $         | Reduce 3      |
(06) |0 E 1                  |+ N / N - N $         | Shift 4       |
(07) |0 E 1 + 4              |N / N - N $           | Shift 3       |
(08) |0 E 1 + 4 N 3          |/ N - N $             | Reduce 6      |
(09) |0 E 1 + 4 T 8          |/ N - N $             | Shift 7       |
(10) |0 E 1 + 4 T 8 / 7      |N - N $               | Shift 11      |
(11) |0 E 1 + 4 T 8 / 7 N 11 |- N $                 | Reduce 5      |
(12) |0 E 1 + 4 T 8          |- N $                 | Reduce 1      |
(13) |0 E 1                  |- N $                 | Shift 5       |
(14) |0 E 1 - 5              |N $                   | Shift 3       |
(15) |0 E 1 - 5 N 3          |$                     | Reduce 6      |
(16) |0 E 1 - 5 T 9          |$                     | Reduce 2      |
(17) |0 E 1                  |$                     | acc           |

Parsing success!

Result: 199.0
```

Reduce x 를 중점적으로 보면,

Step 01: N → T (6)

Step 04: T*N → T (4)

Step 05: T → E (3)

Step 08: N → T (6)

Step 11: T/N → T (5)

Step 12: E+T → E (1)

Step 15: N → T (6)

Step 16: E-T→ E (2)

```
#3

LET'S START!
Enter E
Enter T
Enter T'
Enter T'
epsilon
Exit T'
Exit T'
Exit T
Enter E'
Enter T
Enter T'
Enter T'
epsilon
Exit T'
Exit T'
Exit T
Enter E'
Enter T
Enter T'
epsilon
Exit T'
Exit T
Enter E'
epsilon
Exit E'
Exit E'
Exit E'
Exit E
Recursive-Descent parsing Result: 199.0
```

Enter 를 중점적으로 보면,

E: E → TE'

T: T → NT' | 20 인식 | T' 호출

T': T' → *NT' | * 인식, 10 인식 | T' 호출 → epsilon

E': E' → +TE' | + 인식

T: T → NT' | 30 인식 | T' 호출

T': T' → /NT' | / 인식, 10 인식 | T' 호출 → epsilon

E': E' → - TE' | - 인식

T: T →NT' | 4 인식 | T' 호출 → epsilon

E': epsilon