REPORT

2021-1 Compiler Term-Project

Implement Syntax Analyzer

과목명	컴파일러 01분반
담당교수	김효수 교수님
학과	소프트웨어학부
학년	3학년
학번/이름	20194538 이나혁
	20193418 이하윤
제출일	2021년 6월 5일

Contents

I. Find CFG's Ambiguity

1. 주어진 CFG :

CFG G:

01: CODE → VDECL CODE | FDECL CODE | CDECL CODE| ϵ
02: VDECL → vtype id semi | vtype ASSIGN semi
03: ASSIGN → id assign RHS
04: RHS → EXPR | literal | character | boolstr
05: EXPR → EXPR addsub EXPR | EXPR multdiv EXPR
06: EXPR → lparen EXPR rparen | id | num
07: FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
08: ARG → vtype id MOREARGS | ϵ
09: MOREARGS → comma vtype id MOREARGS | ϵ
10: BLOCK → STMT BLOCK | ϵ
11: STMT → VDECL | ASSIGN semi
12: STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE
13: STMT → while lparen COND rparen lbrace BLOCK rbrace
14: COND → COND comp COND | boolstr
15: ELSE → else lbrace BLOCK rbrace | ϵ
16: RETURN → return RHS semi
17: CDECL → class id lbrace ODECL rbrace
18: ODECL → VDECL ODECL | FDECL ODECL | ϵ

2. 모호함을 제거한 CFG :

CFG G:

00: S → CODE
01: CODE → VDECL CODE | FDECL CODE | CDECL CODE| ϵ
02: VDECL → vtype id semi | vtype ASSIGN semi
03: ASSIGN → id assign RHS
04: RHS → EXPR | literal | character | boolstr
05: EXPR → TERM addsub EXPR | TERM
06: TERM → FACTOR multdiv TERM | FACTOR
07: FACTOR → lparen EXPR rparen | id | num
08: FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
09: ARG → vtype id MOREARGS | ϵ
10: MOREARGS → comma vtype id MOREARGS | ϵ
11: BLOCK → STMT BLOCK | ϵ

12: STMT → VDECL | ASSIGN semi

13: STMT → if lparen COND rparen lbrace BLOCK rbrace ELSE

14: STMT → while lparen COND rparen lbrace BLOCK rbrace

15: COND → FACTOR comp FACTOR | boolstr

16: ELSE → else lbrace BLOCK rbrace | ϵ

17: RETURN → return RHS semi

18: CDECL → class id lbrace ODECL rbrace

19: ODECL → VDECL ODECL | FDECL ODECL | ϵ

- 00: CFG 의 Start 를 S -> CODE 로 수정하였다.
- 05~07: 연산자의 우선순위가 모호해져 우선순위를 가지도록 수정하였다.
- 15: 비교연산자의 우선순위가 모호해져 우선순위를 가지도록 수정하였다.

II. SLR PARSING TABLE

우리는 주어진 웹 사이트를 통해 모호하지 않은 CFG 에 대한 SLR Parsing Table 을 구성했다.
(URL : http://jsmachines.sourceforge.net/machines/slr.html)

- SLR Grammar :

```
S -> CODE
CODE -> VDECL CODE
CODE -> FDECL CODE
CODE -> CDECL CODE
CODE -> epsilon
VDECL -> vtype id semi
VDECL -> vtype ASSIGN semi
ASSIGN -> id assign RHS
RHS -> EXPR
RHS -> literal
RHS -> character
RHS -> boolstr
EXPR -> TERM addsub EXPR
EXPR -> TERM
TERM -> FACTOR multdiv TERM
TERM -> FACTOR
FACTOR -> lparen EXPR rparen
FACTOR -> id
FACTOR -> num
FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
ARG -> vtype id MOREARGS
ARG -> epsilon
MOREARGS -> comma vtype id MOREARGS
MOREARGS -> epsilon
BLOCK -> STMT BLOCK
BLOCK -> epsilon
STMT -> VDECL
STMT -> ASSIGN semi
STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE
STMT -> while lparen COND rparen lbrace BLOCK rbrace
COND -> FACTOR comp FACTOR
COND -> boolstr
ELSE -> else lbrace BLOCK rbrace
ELSE -> epsilon
RETURN -> return RHS semi
CDECL -> class id lbrace ODECL rbrace
ODECL -> VDECL ODECL
ODECL -> FDECL ODECL
ODECL -> epsilon
```

- SLR Parsing Table

State	ACTION																				GOTO														
	vtype	id	Semi	assign	literal	character	boolstr	addsub	multdiv	lparen	rparen	num	lbrace	rbrace	comma	if	while	comp	else	return	class	$	S	CODE	VDECL	ASSIGN	RHS	EXPR	TERM	FACTOR	FDECL	ARG	MOREARGS	BLOCK	STMT	COND	ELSE	RETURN	CDECL	ODECL
0	S5																			S6	R4		1	2						3								4	
1																					acc																			
2	S5																			S6	R4		7	2						3								4	
3	S5																			S6	R4		8	2						3								4	
4	S5																			S6	R4		9	2						3								4	
5		S10																							11															
6		S12																																						
7																					R1																			
8																					R2																			
9																					R3																			
10			S13	S15						S14																														
11			S16																																					
12													S17																											
13	R5	R5												R5		R5	R5			R5	R5	R5																		
14	S19										R21																				18									
15		S28			S22	S23	S24			S27		S29															20	21	25	26										
16	R6	R6												R6		R6	R6			R6	R6	R6																		
17	S5													R38										31							32									30
18											S33																													
19		S34																																						
20			R7																																					
21			R8																																					
22			R9																																					
23			R10																																					
24			R11																																					
25			R13					S35			R13																													
26			R15					R15	S36		R15																													
27		S28								S27		S29															37	25	26											
28			R17					R17	R17		R17							R17																						
29			R18					R18	R18		R18							R18																						
30														S38																										
31	S5													R38										31							32									39
32	S5													R38										31							32									40
33													S41																											
34											R23				S43																		42							
35		S28								S27		S29															44	25	26											
36		S28								S27		S29																45	26											
37											S46																													
38	R35																			R35	R35																		
39														R36																										
40														R37																										
41	S53	S54												R25		S51	S52			R25				49	50									47	48					
42											R20																													
43	S55																																							
44			R12								R12																													
45			R14					R14			R14																													
46			R16					R16	R16		R16							R16																						
47																				S57																		56		
48	S53	S54												R25		S51	S52			R25				49	50									58	48					
49	R26	R26												R26		R26	R26			R26																				
50			S59																																					
51										S60																													
52										S61																													
53		S62																							11														
54				S15																																				
55		S63																																					
56														S64																										
57		S28			S22	S23	S24			S27		S29															65	21	25	26										
58														R24						R24																				
59	R27	R27												R27		R27	R27			R27																				
60		S28					S68			S27		S29																		67						66				
61		S28					S68			S27		S29																		67						69				
62			S13	S15																																				
63											R23				S43																70								
64	R19													R19							R19	R19																		
65			S71																																					
66											S72																													
67																		S73																						
68											R31																													
69											S74																													
70											R22																													
71														R34																										
72													S75																											
73		S28								S27		S29																		76										
74													S77																											
75	S53	S54												R25		S51	S52			R25				49	50									78	48					
76											R30																													
77	S53	S54												R25		S51	S52			R25				49	50									79	48					
78														S80																										
79														S81																										
80	R33	R33												R33		R33	R33		S83	R33																		82		
81	R29	R29												R29		R29	R29			R29																				
82	R28	R28												R28		R28	R28			R28																				
83													S84																										
84	S53	S54												R25		S51	S52			R25				49	50									85	48					
85														S86																										
86	R32	R32												R32		R32	R32			R32																				

III. Implementation

0. Developing Environmemt

Name	OS	Language	IDE
이나혁	Mac OS 11 Big Sur (Intel Processor)	Python3.6	PyCharm
이하윤	Mac OS 11 Big Sur (M1 Silicon)	Python3.8	PyCharm

1. Definition of Rules

```
RULE = [
    ['S', 'CODE'],
    ['CODE', 'VDECL CODE'],
    ['CODE', 'FDECL CODE'],
    ['CODE', 'CDECL CODE'],
    ['CODE', 'epsilon'],
    ['VDECL', 'vtype id semi'],
    ['VDECL', 'vtype ASSIGN semi'],
    ['ASSIGN', 'id assign RHS'],
    ['RHS', 'EXPR'],
    ['RHS', 'literal'],
    ['RHS', 'character'],
    ['RHS', 'boolstr'],
    ['EXPR', 'TERM addsub EXPR'],
    ['EXPR', 'TERM'],
    ['TERM', 'FACTOR multdiv TERM'],
    ['TERM', 'FACTOR'],
    ['FACTOR', 'lparen EXPR rparen'],
    ['FACTOR', 'id'],
    ['FACTOR', 'num'],
    ['FDECL', 'vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace'],
    ['ARG', 'vtype id MOREARGS'],
    ['ARG', 'epsilon'],
    ['MOREARGS', 'comma vtype id MOREARGS'],
    ['MOREARGS', 'epsilon'],
    ['BLOCK', 'STMT BLOCK'],
    ['BLOCK', 'epsilon'],
    ['STMT', 'VDECL'],
    ['STMT', 'ASSIGN semi'],
    ['STMT', 'if lparen COND rparen lbrace BLOCK rbrace ELSE'],
    ['STMT', 'while lparen COND rparen lbrace BLOCK rbrace'],
```

```
    ['COND', 'FACTOR comp FACTOR'],
    ['COND', 'boolstr'],
    ['ELSE', 'else lbrace BLOCK rbrace'],
    ["ELSE', 'epsilon"],
    ['RETURN', 'return RHS semi'],
    ['CDECL', 'class id lbrace ODECL rbrace'],
    ['ODECL', 'VDECL ODECL'],
    ['ODECL', 'FDECL ODECL'],
    ['ODECL', 'epsilon']
]
```

위와 같이 정의한 Rule 을 리스트 자료형을 사용해 정의했다.

2. Definition of SLR Parsing Table

```
SLR_TABLE = [
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 1, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'$': 'ACC', },
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 7, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 8, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'vtype': 'S5', 'class': 'S6', '$': 'R4', 'CODE': 9, 'VDECL': 2, 'FDECL': 3, 'CDECL': 4, },
    {'id': 'S10', 'ASSIGN': 11, },
    {'id': 'S12', },
    {'$': 'R1', },
    {'$': 'R2', },
    {'$': 'R3', },
    {'semi': 'S13', 'assign': 'S15', 'lparen': 'S14', },
    {'semi': 'S16', },
    {'lbrace': 'S17', },
    {'vtype': 'R5', 'id': 'R5', 'rbrace': 'R5', 'if': 'R5', 'while': 'R5', 'return': 'R5', 'class': 'R5', '$': 'R5', },
    {'vtype': 'S19', 'rparen': 'R21', 'ARG': 18, },
    {'id': 'S28', 'literal': 'S22', 'character': 'S23', 'boolstr': 'S24', 'lparen': 'S27', 'num': 'S29', 'RHS': 20,
     'EXPR': 21, 'TERM': 25, 'FACTOR': 26, },
    {'vtype': 'R6', 'id': 'R6', 'rbrace': 'R6', 'if': 'R6', 'while': 'R6', 'return': 'R6', 'class': 'R6', '$': 'R6', },
    {'vtype': 'S5', 'rbrace': 'R38', 'VDECL': 31, 'FDECL': 32, 'ODECL': 30, },
    {'rparen': 'S33', },
    {'id': 'S34', },
    {'semi': 'R7', },
    {'semi': 'R8', },
    {'semi': 'R9', },
    {'semi': 'R10', },
    {'semi': 'R11', },
    {'semi': 'R13', 'addsub': 'S35', 'rparen': 'R13', },
    {'semi': 'R15', 'addsub': 'R15', 'multdiv': 'S36', 'rparen': 'R15', },
    {'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'EXPR': 37, 'TERM': 25, 'FACTOR': 26, },
    {'semi': 'R17', 'addsub': 'R17', 'multdiv': 'R17', 'rparen': 'R17', 'comp': 'R17', },
    {'semi': 'R18', 'addsub': 'R18', 'multdiv': 'R18', 'rparen': 'R18', 'comp': 'R18', },
    {'rbrace': 'S38', },
    {'vtype': 'S5', 'rbrace': 'R38', 'VDECL': 31, 'FDECL': 32, 'ODECL': 39, },
    {'vtype': 'S5', 'rbrace': 'R38', 'VDECL': 31, 'FDECL': 32, 'ODECL': 40, },
    {'lbrace': 'S41', },
    {'rparen': 'R23', 'comma': 'S43', 'MOREARGS': 42, },
    {'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'EXPR': 44, 'TERM': 25, 'FACTOR': 26, },
    {'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'TERM': 45, 'FACTOR': 26, },
    {'rparen': 'S46', },
    {'vtype': 'R35', 'class': 'R35', '$': 'R35', },
    {'rbrace': 'R36', },
    {'rbrace': 'R37', },
    {'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49,
     'ASSIGN': 50, 'BLOCK': 47, 'STMT': 48, },
    {'rparen': 'R20', },
    {'vtype': 'S55', },
    {'semi': 'R12', 'rparen': 'R12', },
```

```
    {'semi': 'R14', 'addsub': 'R14', 'rparen': 'R14', },
    {'semi': 'R16', 'addsub': 'R16', 'multdiv': 'R16', 'rparen': 'R16', 'comp': 'R16', },
    {'return': 'S57', 'RETURN': 56, },
    {'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49,
     'ASSIGN': 50, 'BLOCK': 58, 'STMT': 48, },
    {'vtype': 'R26', 'id': 'R26', 'rbrace': 'R26', 'if': 'R26', 'while': 'R26', 'return': 'R26', },
    {'semi': 'S59', },
    {'lparen': 'S60', },
    {'lparen': 'S61', },
    {'id': 'S62', 'ASSIGN': 11, },
    {'assign': 'S15', },
    {'id': 'S63', },
    {'rbrace': 'S64', },
    {'id': 'S28', 'literal': 'S22', 'character': 'S23', 'boolstr': 'S24', 'lparen': 'S27', 'num': 'S29', 'RHS': 65,
     'EXPR': 21, 'TERM': 25, 'FACTOR': 26, },
    {'rbrace': 'R24', 'return': 'R24', },
    {'vtype': 'R27', 'id': 'R27', 'rbrace': 'R27', 'if': 'R27', 'while': 'R27', 'return': 'R27', },
    {'id': 'S28', 'boolstr': 'S68', 'lparen': 'S27', 'num': 'S29', 'FACTOR': 67, 'COND': 66, },
    {'id': 'S28', 'boolstr': 'S68', 'lparen': 'S27', 'num': 'S29', 'FACTOR': 67, 'COND': 69, },
    {'semi': 'S13', 'assign': 'S15', },
    {'rparen': 'R23', 'comma': 'S43', 'MOREARGS': 70, },
    {'vtype': 'R19', 'rbrace': 'R19', 'class': 'R19', '$': 'R19', },
    {'semi': 'S71', },
    {'rparen': 'S72', },
    {'comp': 'S73', },
    {'rparen': 'R31', },
    {'rparen': 'S74', },
    {'rparen': 'R22', },
    {'rbrace': 'R34', },
    {'lbrace': 'S75', },
    {'id': 'S28', 'lparen': 'S27', 'num': 'S29', 'FACTOR': 76, },
    {'lbrace': 'S77', },
    {'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49,
     'ASSIGN': 50, 'BLOCK': 78, 'STMT': 48, },
    {'rparen': 'R30', },
    {'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49,
     'ASSIGN': 50, 'BLOCK': 79, 'STMT': 48, },
    {'rbrace': 'S80', },
    {'rbrace': 'S81', },
    {'vtype': 'R33', 'id': 'R33', 'rbrace': 'R33', 'if': 'R33', 'while': 'R33', 'else': 'S83', 'return': 'R33',
     'ELSE': 82, },
    {'vtype': 'R29', 'id': 'R29', 'rbrace': 'R29', 'if': 'R29', 'while': 'R29', 'return': 'R29', },
    {'vtype': 'R28', 'id': 'R28', 'rbrace': 'R28', 'if': 'R28', 'while': 'R28', 'return': 'R28', },
    {'lbrace': 'S84', },
    {'vtype': 'S53', 'id': 'S54', 'rbrace': 'R25', 'if': 'S51', 'while': 'S52', 'return': 'R25', 'VDECL': 49,
     'ASSIGN': 50, 'BLOCK': 85, 'STMT': 48, },
    {'rbrace': 'S86', },
    {'vtype': 'R32', 'id': 'R32', 'rbrace': 'R32', 'if': 'R32', 'while': 'R32', 'return': 'R32', },
    ]
```

위와 같이 딕셔너리형 리스트를 사용해 SLR Parsing Table 을 정의했다.

3. SLR Stack

```
def parser():
    if len(terminal_list) == 1:
        return True, ''

    SLR_stack = [0]
    splitter = 0
    error_line = 1
    check = 0
```

```
    while True:
        next_input_symbol = terminal_list[splitter]
        check += 1
        current_state = SLR_stack[-1]
```

parser()함수를 정의했으며, SLR Stack, splitter 를 선언했다. 초기 stack 에는 0 이 들어있으며, splitter 는 0 에서 시작한다. check 와 error_line 은 reject 시 에러 메시지 출력에 사용될 변수이다. current_state 는 항상 SLR stack 의 마지막 원소와 같다.

4. Shift

파서는 현재 state 에서 다음 입력 symbol 을 읽어 Shift, Reduce, GOTO, Accept 중에 한 가지 명령을 수행한다.

```
# 판단할 수 없는 symbol 이 있으면 error
if next_input_symbol not in SLR_TABLE[current_state].keys():
    error = "[REJECTED] Error in line " + str(error_line) + ", " + error_checker[error_line-1]
    print(error)
    return False, error
```

SLR Table 에 input 으로 정의되지 않은 symbol 이 있으면 에러이므로, 가장 먼저 SLR table 에 해당 symbol 이 존재하는지 판단한다.

그 후, SLR Table 의 이동 상태가 S 로 시작하면 Shift 를 수행하도록 한다.

```
# shift
if SLR_TABLE[current_state][next_input_symbol][0] == 'S':
    splitter += 1
    error_line += 1
    SLR_stack.append(int(SLR_TABLE[current_state][next_input_symbol][1:]))
```

Shift 는 symbol 을 읽고 지나가는 것이므로 splitter 에 1 을 더하고, SLR Stack 에 이동 상태의 숫자(State)를 삽입한다.

5. Reduce & GOTO

1. Reduce

SLR Table 의 이동 상태가 R 로 시작하면 Reduce 를 수행하도록 한다.

```
# Reduce
elif SLR_TABLE[current_state][next_input_symbol][0] == 'R':
    table_num = SLR_TABLE[current_state][next_input_symbol][1:]

    non_termi = RULE[int(table_num)][0]   # ex) CODE → epsilon 에서 VDECL 의미
    termi = RULE[int(table_num)][1]   # ex) CODE → epsilon 에서 epsilon 의미
    termi_length = len(str(termi).split(" "))   # 룰의 어절 개수   ex) VDECL → vtype id semi : 3

    for i in range(termi_length):
        if termi != 'epsilon':
            SLR_stack.pop()
            terminal_list.pop(splitter-i-1)

    if non_termi not in SLR_TABLE[SLR_stack[-1]].keys():
        error = "[REJECTED] Error in line " + str(error_line) + ", " + error_checker[error_line - 1]
```

```
        print(error)
        return False, error
```

Reduce 는 정의한 Rule 에 따라 terminal 혹은 non-terminal 을 non-terminal 로 줄이는 과정이다. 만약 SLR Table 에서 읽은 Reduce 의 number 에 해당하는 Rule 이 epsilon 을 non-terminal 로 바꾸는 것이 아니라면 rule 의 어절의 개수만큼 SLR stack 에서 pop 을 실행한다. (ex. Rule 이 VDECL -> vtype id semi 라면 SLR Stack 에서 pop 을 3 번 실행한다.)

만약 정의한 rule 에서 읽은 non-terminal 이 SLR Table 에 없다면 에러이므로 에러 메시지를 출력하도록 한다.

```
# GOTO
SLR_stack.append(SLR_TABLE[SLR_stack[-1]][non_termi])

if termi != 'epsilon':
    splitter = splitter - termi_length + 1
else:
    splitter += 1

terminal_list.insert(splitter-1, non_termi)
```

Reduce 를 실행하게 되면 non-terminal 을 terminal_list 에 삽입하게 된다. 이 non-terminal 을 SLR Table 에서 읽은 결과를 SLR Stack 에 삽입한다. (GOTO 과정)

Rule 에 따라 terminal_list 의 terminal 과 non-terminal 을 줄이는 개수가 달라지므로, 그에 맞게 spliter 위치를 조절한다.

그 후 Reduce 한 Rule 의 non-terminal 을 terminal list 에 삽입한다.

6. Accept

```
if current_state == 1 and next_input_symbol == '$':
    print("ACCEPTED!!!")
    return True
```

현재 State 가 1 이고 next_input_symbol 이 '$'이면 Accept 되는 경우이므로 command 창에 Accept 메시지를 출력한다.

7. Others

1. lexical analyzer

```
### 결과를 새로운 파일에 출력 및 저장하는 함수
def save_result():
    global token_value
    global token_key
    # Output File 에 대한 이름은 '<input_file_name>_output.txt'로 저장됨
    save = open(file_path + '_output.txt', 'w')
    for i, j in zip(token_value, token_key):
        # White Space 를 제외하고 저장
        if j != 'WHITE_SPACE':
```

```
                #save.write("<" + j + ", " + i + ">\n")
                save.write(j + "\n")
                # print("<"+j+", i+">")
    print("Successfully token list saved")
    save.close
```

Project 1 에서 설계했던 lexical analyzer 를 syntax analyzer 에 사용하기 위해 lexical anaylzer 의 출력 형식을 token 의 분석 결과만 출력하도록 수정하였다.

2. 전체 process

Input file 은 Command 에서 Input File 명을 받아와 읽게 된다.

```
# Command 내 인자 개수 확인
if len(sys.argv) != 2:
    print("Insufficient arguments")
    sys.exit()

# input File 열기
file_path = sys.argv[1]
f = open(file_path, 'r')

END_MARK = '$'
terminal_list = []

lines = f.readlines()
for line in lines:
    terminal = line.split()[0]
    terminal_list.append(terminal)

terminal_list.append(END_MARK)
error_checker = copy.copy(terminal_list)

parser()
```

파일을 읽은 후, 위에서 정의한 parser()함수를 실행한다. parser() 함수는 input terminal list 를 shift, reduce 작업을 반복해 최종적으로 accept 와 reject 을 판단하므로 True 를 반환하거나 False 를 반환할 때까지 shift 와 reduce 를 반복한다.

IV. Experiment

1. Aceept

- Case 1 :

Input.java	Input.java_output.txt

class Person {
  int a = 1;
}
	class
	id
	lbrace
	vtype
	id
	assign
	num
	semi
	rbrace

Result:

```
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python lexical_analyzer.py input.java
Successfully token list saved
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python syntax_analyzer.py input.java_output.txt
ACCEPTED!!!
```

- Case 2 :

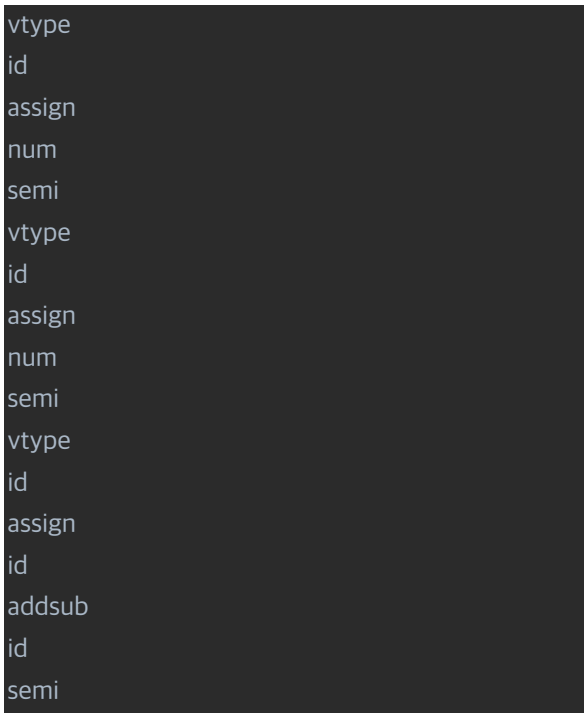
Input2.java	Input2.java_output.txt

float a = 0;
 	
class Person {
  int a = 1;
}
	vtype
	id
	assign
	num
	semi
	class
	id
	lbrace
	vtype
	id
	assign
	num
	semi
	rbrace

Result:

```
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python lexical_analyzer.py input2.java
Successfully token list saved
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python syntax_analyzer.py input2.java_output.txt
ACCEPTED!!!
```

- Case 3 :

Input3.java	Input3.java_output.txt
	

Result:

```
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python lexical_analyzer.py input3.java
Successfully token list saved
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python syntax_analyzer.py input3.java_output.txt
ACCEPTED!!!
```

2. Reject

- Case 1 :

Input4.java	Input4.java_output.txt
`class Person {` `  int a = 1;`	class id lbrace vtype id assign num semi

Result:

```
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python lexical_analyzer.py input4.java
Successfully token list saved
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python syntax_analyzer.py input4.java_output.txt
[REJECTED] Error in line 9, $
```

- Case 2 :

Input5.java	Input5.java_output.txt

`float a = 0;`
 	
`class Person{`
`  int  = 1;`
`}`
	vtype
	id
	assign
	num
	semi
	class
	id
	lbrace
	vtype
	assign
	num
	semi
	rbrace

Result:

```
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python lexical_analyzer.py input5.java
Successfully token list saved
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python syntax_analyzer.py input5.java_output.txt
[REJECTED] Error in line 10, assign
```

- Case 3 :

Input6.java	Input6.java_output.txt

`int a = 3;`
`int b = 5;`
`a+b = int c;`
	vtype
	id
	assign
	num
	semi
	vtype
	id
	assign
	num
	semi
	id
	addsub
	id
	assign
	vtype
	id
	semi

Result:

```
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python lexical_analyzer.py input6.java
Successfully token list saved
(base) hayunlee@Hayunui-MacBookAir syntax analyzer % python syntax_analyzer.py input6.java_output.txt
[REJECTED] Error in line 11, id
```