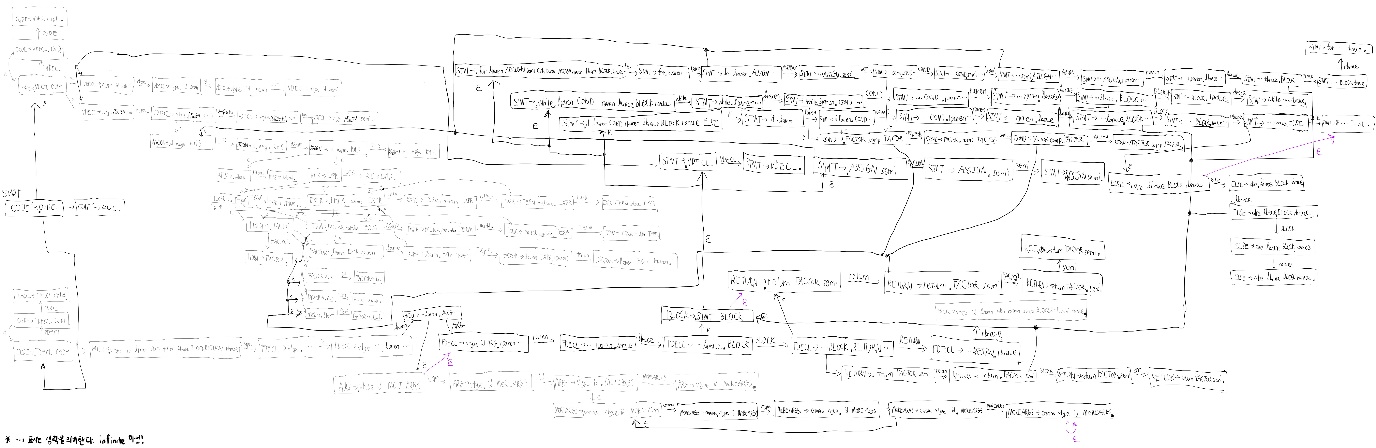
COMPILER PROJECT 2

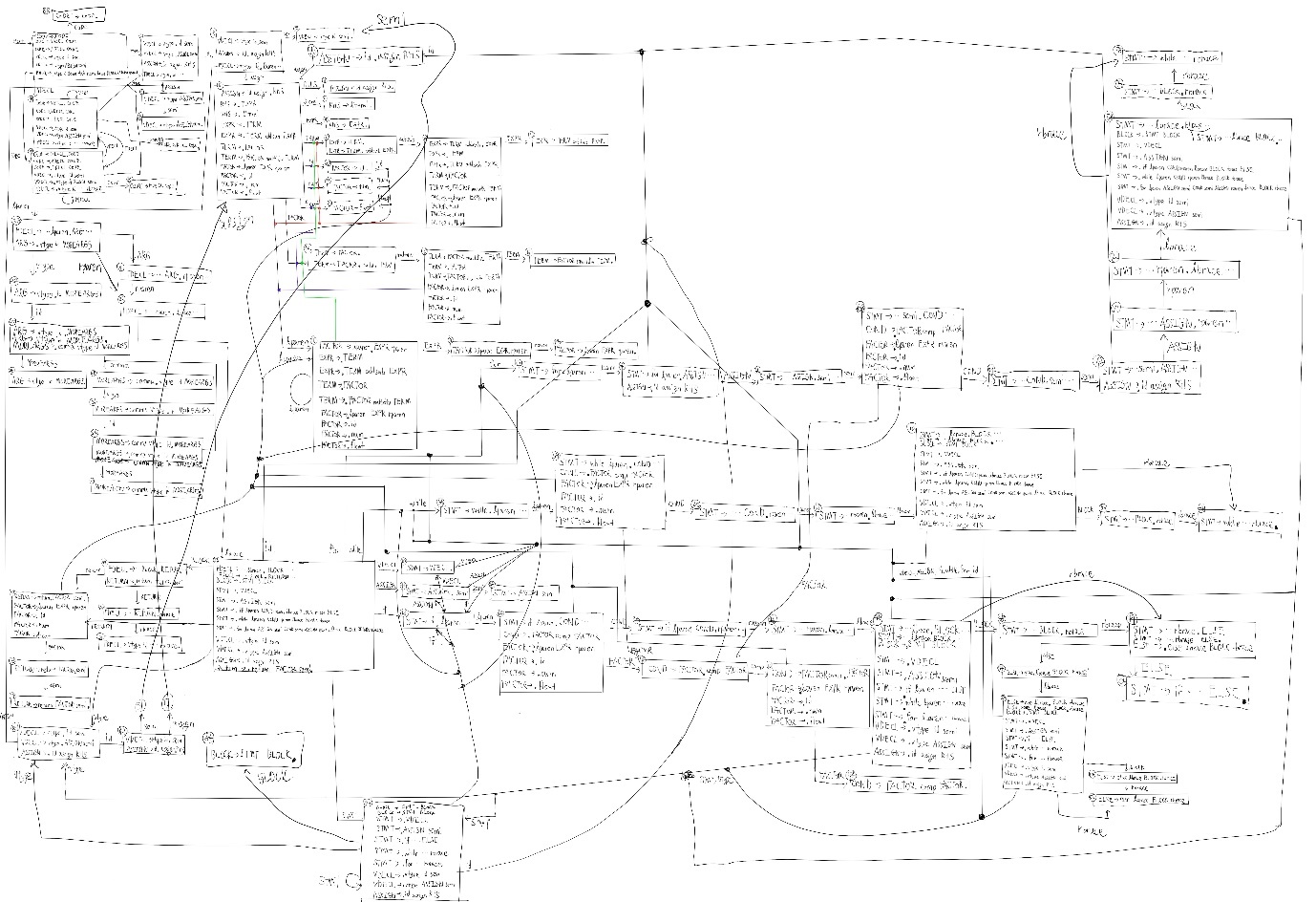
20155394 김웅수

INDEX

1. NFA
2. DFA
3. FOLLOW SETS
4. SLR PARSING TABLE
5. CODE EXPLAIN
6. RUN CODE

1. NFA



2. DFA

BLOCK, ARG, MOREARGS, ELSE 등 뒤에 epsilon이 올 수 있는 경우도 포함하였다.

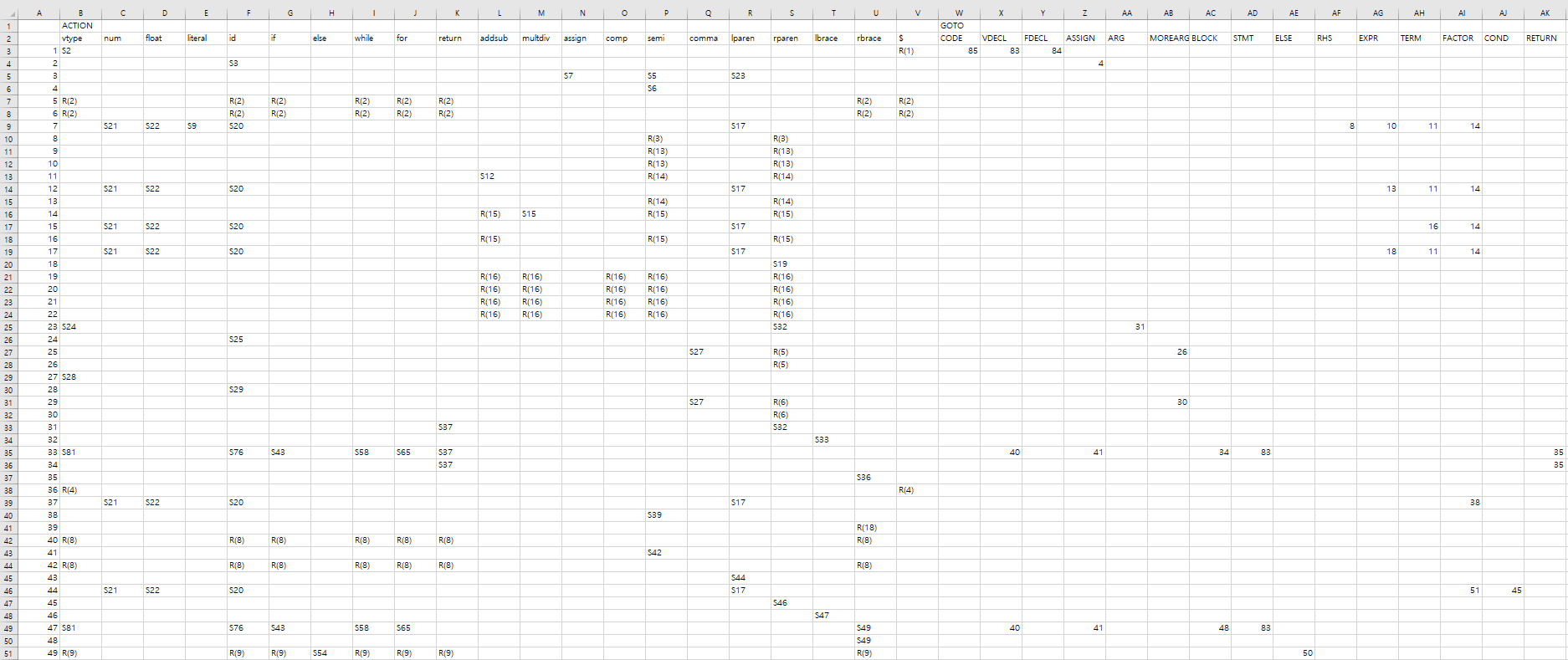
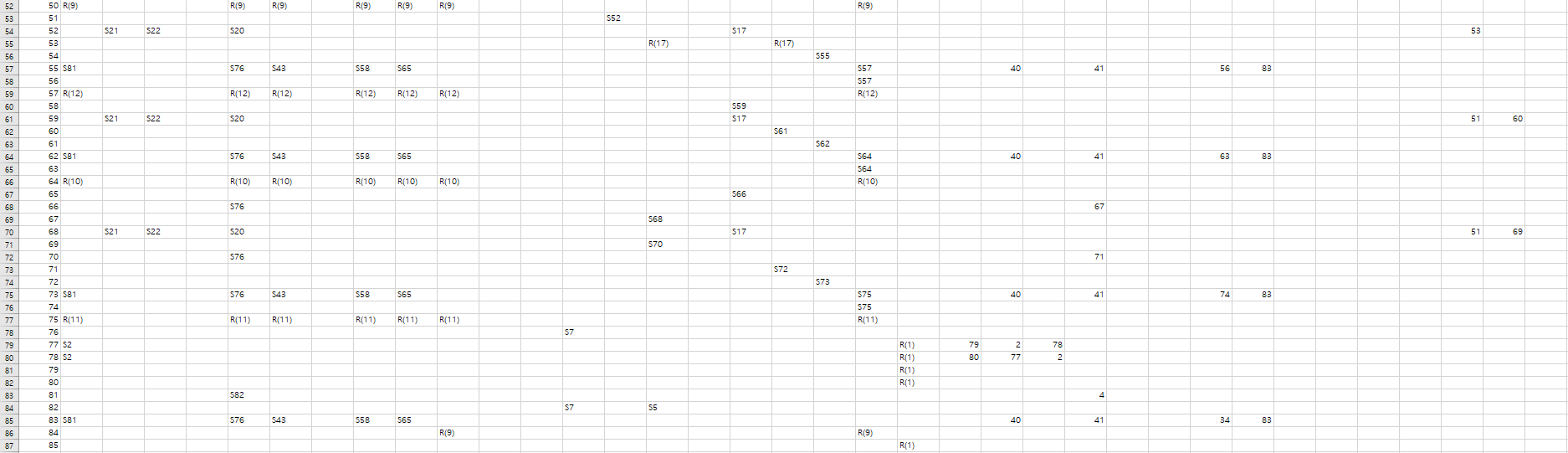
85번 state CODE’ 에 도달할 경우 accepted 상태라고 하였다.

3. FOLLOW SETS



FIRST SET을 구한 후 FOLLOW SET을 구한다.

4. SLR PARSING TABLE

일부 epsilon이 나올 수 있는 non-terminal을 고려하여 작성하였다.

5. CODE EXPLAIN

위 SLR PARSING TABLE을 기반으로 코드를 작성하였다.

- 각 STATE별 ACTION – SHIFT, REDUCE 실행

- REDUCE를 하기 위한 CFG – REDUCE 후 나온 NON-TERMINAL을 기반으로 GOTO 실행

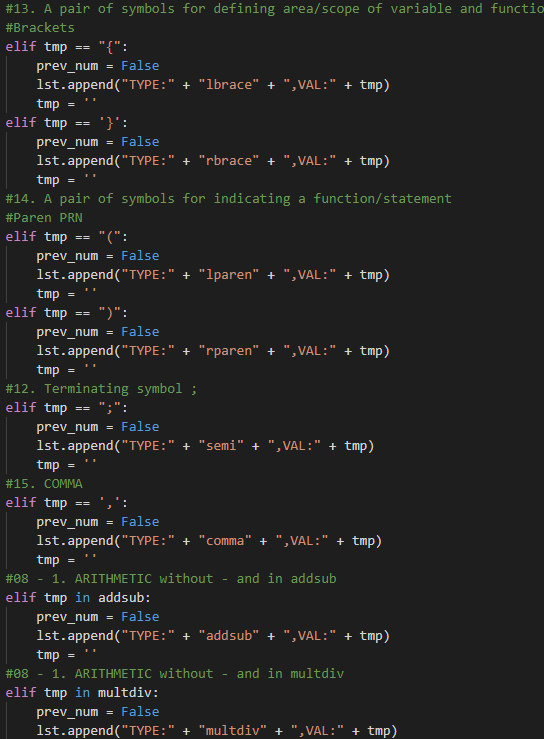
으로 크게 나누어져서 코드는 시행되었다.

1. 지난 코드 변경점

저번 TERM PROJECT 때에 제출하였던 lexical analyzer와 이번 syntax analyzer에서의 terminal name과 맞지 않아 그 부분을 수정하였다.



기존의 vtype의 TYPE에는 int, char, bool, float이 그대로 type에 들어갔으나 이번에는 묶어서 vtype으로 정의되었다.



그 외 각종 TYPE명을 수정하였다.

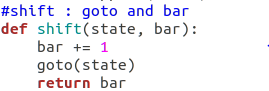
1. 초기화



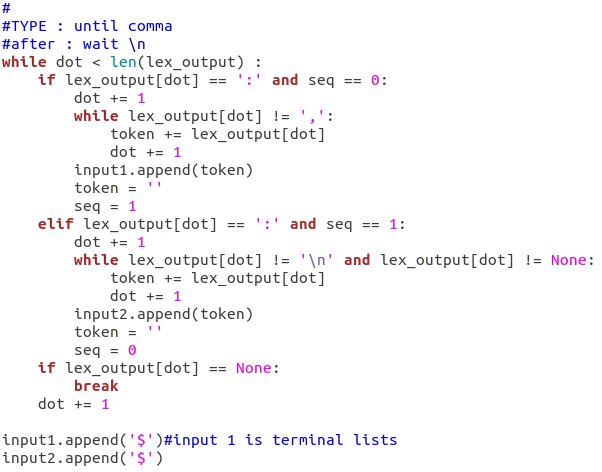
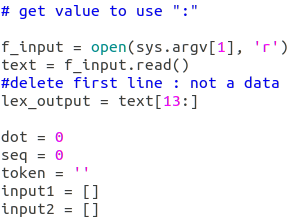
수정한 TYPE을 기반으로 TERMINAL과 NON-TERMINAL을 구분하여 정의하였다. (float은 사용하지 않아 그대로 TYPE으로서 사용하였지만, if, else 등 많이 사용하는 keywords들은 string형태로 매번 지정하여 사용하였다.)



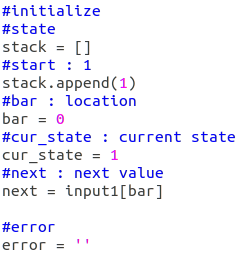
goto는 SLR PARSING에서의 goto와 동일하게 만들었다. state를 goto하게 되면 그 state값이 stack에 들어오게 되며, pointer인 bar는 움직이지 않는다.



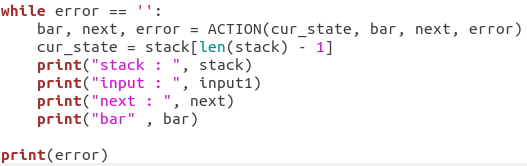
shift는 terminal값에 따라 다음에 이동할 state가 나오게되면 stack에 넣어주고 pointer인 bar를 한 칸 옮겨준다.



“:”을 기준으로 무슨 type인지, 그 값은 무엇인지를 받는다. 이 때 기존의 코드와 동일하게 이번에는 lexical analyzer의 output 파일명인 “output.txt”를 sys.argv[1]을 통해 입력받는다. 그리고 그 값들을 TYPE은 input1, VALUE는 input2 리스트에 넣은 후 끝을 알리는 문자 $을 넣어준다.

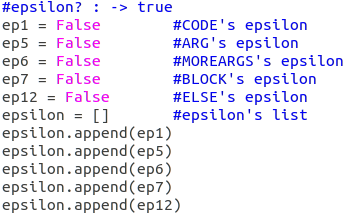


stack은 state로 정의하였고, 초기에 stack엔 state1이 들어갈 것이므로 1을 append해준다. 그리고 현재 state를 가리키는 cur\_state에는 1을 넣어주었고, next는 bar라는 포인터가 현재 가리키는 값이다. error는 기존에는 아무것도 없도록 설정한다.



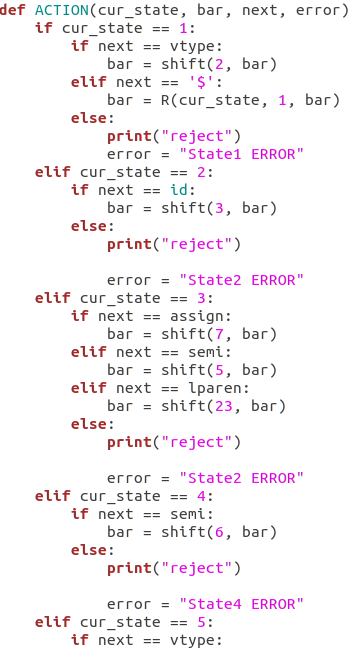
error에 무슨 값이 뜰 때까지 while문을 돌린다. error에 들어가는 값은 어떤 state에서 에러가 발생하였는지 혹은 accepted되었다면 accepted가 들어간다.

어떤 식으로 시연되는지 보여주기 위해 stack, type input, next, bar를 한 ACTION이 끝날 때마다 print한다. (시연 후에는 while 내의 출력들은 모두 지웠다.)

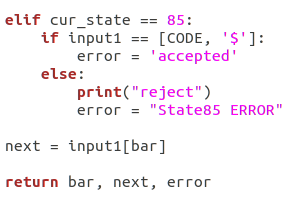


epsilon이 올 수 있는 값들이 있기 때문에 입실론이 아닐 경우 False, epsilon일 경우 true를 저장하는 리스트이다.

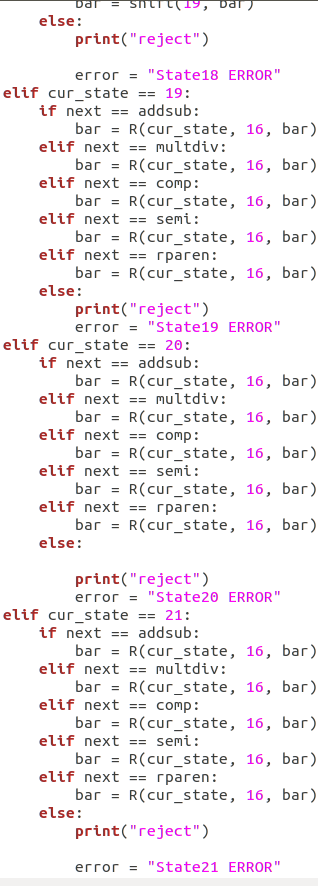
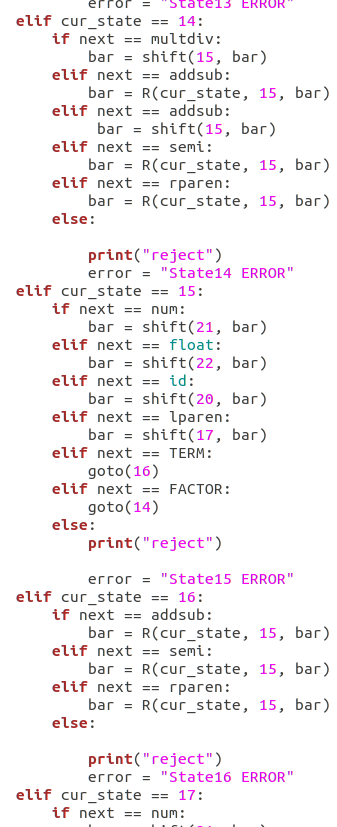
1. ACTION



ACTION이라는 함수 내에는 총 85개의 state가 있기 때문에 각 state별로 맞는 input이 들어갔는지 확인한 후에 shift를 할지, reduce(R() 이라는 함수가 reduce함수이다.)를 할지를 결정한다. 만약 맞지 않는 값이 들어오게 된다면 reject를 띄운 후 error string에 값을 넣어 while문을 종료하게 한다.



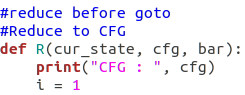
shift 또는 reduce로 값이 바뀌었을 것이기 때문에 85 state로 왔을 경우 기존 input1이라는 기존에 type들이 들어있고 SLR PARSING이 이루어진 stack에 CODE$ 만 남아있을 경우 error가 없이 accepted 되었다는 메시지를 띄울 수 있도록 하며 while문을 종료한다. 그리고 각 state의 종료마다 변화될 수 있는 값들인 bar(pointer), next(pointer가 가리키는 값), error(error가 떴는지 아닌지)를 return한다. 또 reduce될 경우 bar위치에 해당하는 값이 아닌 non-terminal값을 next에 넣게 되기 때문에 next값을 bar위치에 해당하는 값으로 바꿔준다. shift의 경우도 next가 이전 bar값으로 되어있으므로 업데이트 해준다.



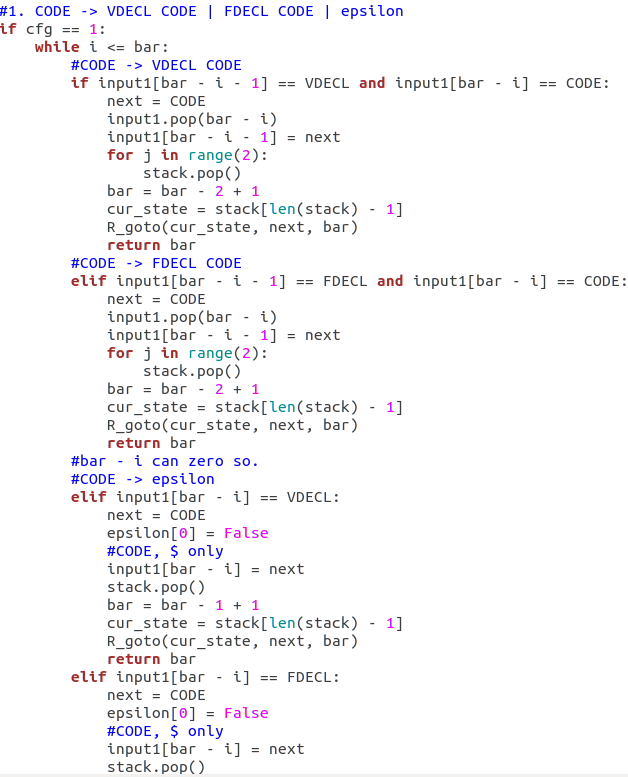
계속 이런 것들이 반복된다. 자세한 것은 DFA그림 혹은 SLR TABLE 참조.

1. reduce (REDUCE 후 GOTO까지 같이 실행한다)

shift, goto와는 다르게 reduce는 따로 분리하여 설명하는 이유는 reduce가 CFG를 따라야 하기 때문이다. reduce함수가 실행될 경우 이루어질 경우 CFG에 맞는 값이 현재 가리키고 있는 포인터 bar 이전까지에 있는지 확인하여야 한다.

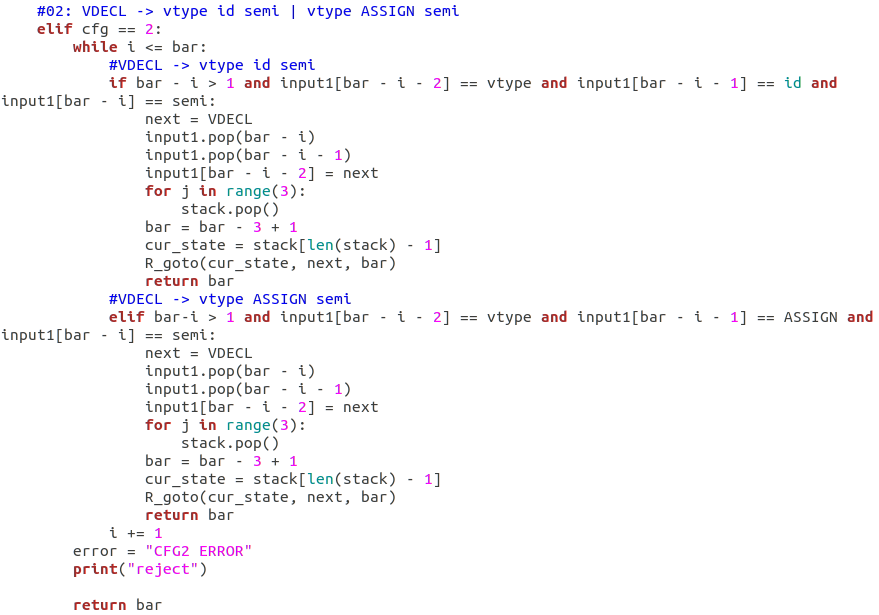


reduce는 어떤 CFG Rule을 실행할지 cfg라는 값을 받는다. i는 현재 bar값에서 -i를 해줘서 bar 이전의 값들이 CFG Rule에 맞는게 있는지 확인해주기 위한 값이다.



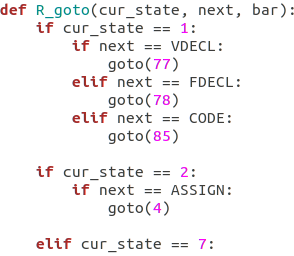
01: CODE → VDECL CODE | FDECL CODE | ϵ

CFG Rule 1은 VDECL CODE, FDECL CODE가 CODE로 Reduce되는 것이다. 그리고 뒤의 CODE가 epsilon일 경우를 생각해 VDECL, FDECL만 각각 있을 경우도 생각한다. 그럴 경우 input1에 들어있는 리스트에서 해당 값에 VDECL CODE 등을 CODE로 바꿔주고, stack에 들어있는 state도 이 state까지 오기 위해 이동한 만큼 pop해준다. 오른쪽에서부터 순서대로 찾기 위해 bar-i로 구한다. i값이 증가함에 따라 왼쪽을 찾는다. input값이 아무것도 들어오지 않은 경우를 대비해 만약 input1의 길이가 1(초기화 때 넣어준 $)밖에 없을 경우를 생각하여 만약 그런 경우 바로 accept하게 될 수 있도록한다.

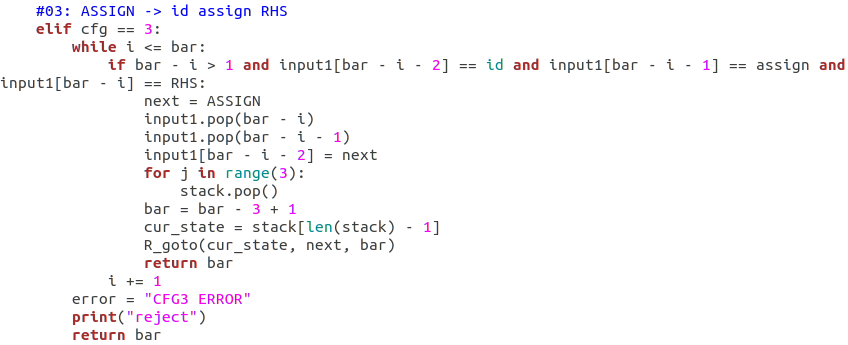


02: VDECL → vtype id semi | vtype ASSIGN semi

위와 동일하다. vtype id semi를 모두 pop한 후 VDECL을 input1에 넣어주고 bar를 움직여준다. bar는 input1에 마지막으로 VDECL을 넣어줬으므로 pop된 수 -1만큼만 움직여준다. state도 같이 pop해준다. pop된 만큼 stack에 들어있는 state를 pop해준 후 pop된 뒤의 current state에 VDECL이 올 경우 어떻게 goto해야 하는지 알기 위해 R\_goto()함수에 각 조건들을 넣은 채로 실행한다. bar-i가 최소한 0보다는 커야 하므로 bar-i>1이라는 조건을 넣어준다. 맞는 값이 없다면 reject를 출력하고 error에 CFG2에서 ERROR가 발생하였다는 것을 넣는다.

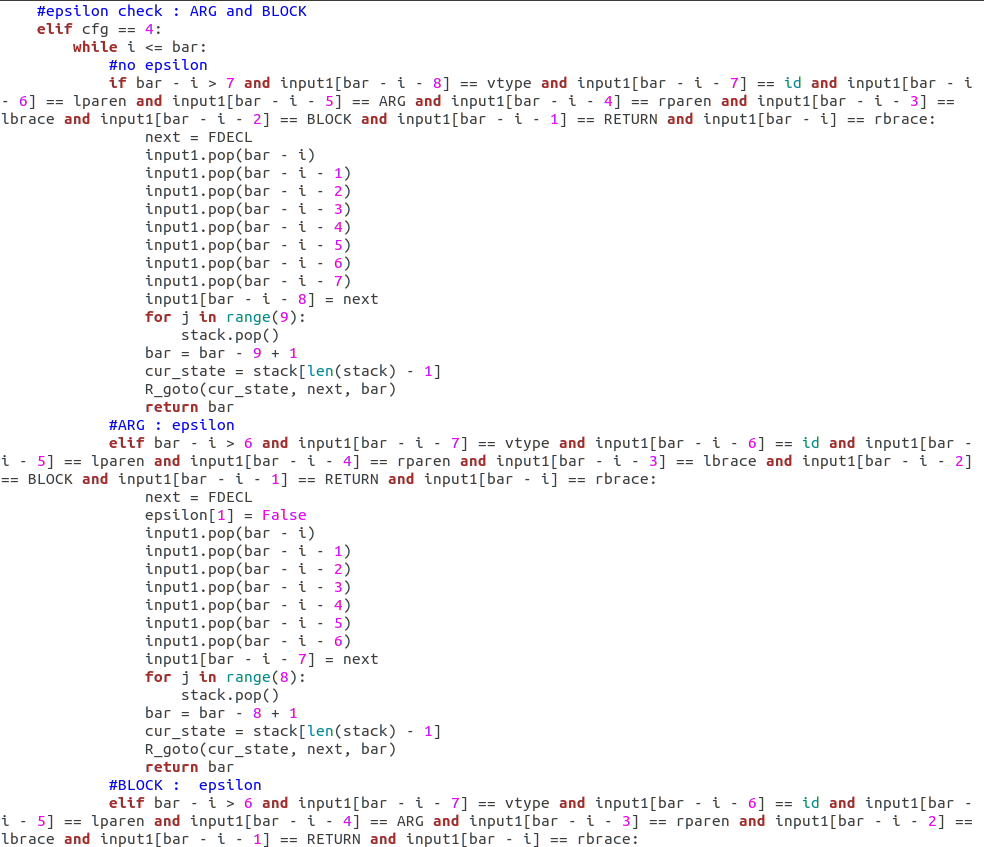


R\_goto 함수는 Reduce 후 pop된 후 가장 최신의 current state와 나타난 non-terminal값을 기반으로 어디로 state를 goto를 할지 적어놓은 함수이다. SLR PARSING TABLE의 모든 state별 goto를 넣은 함수.



03: ASSIGN → id assign RHS

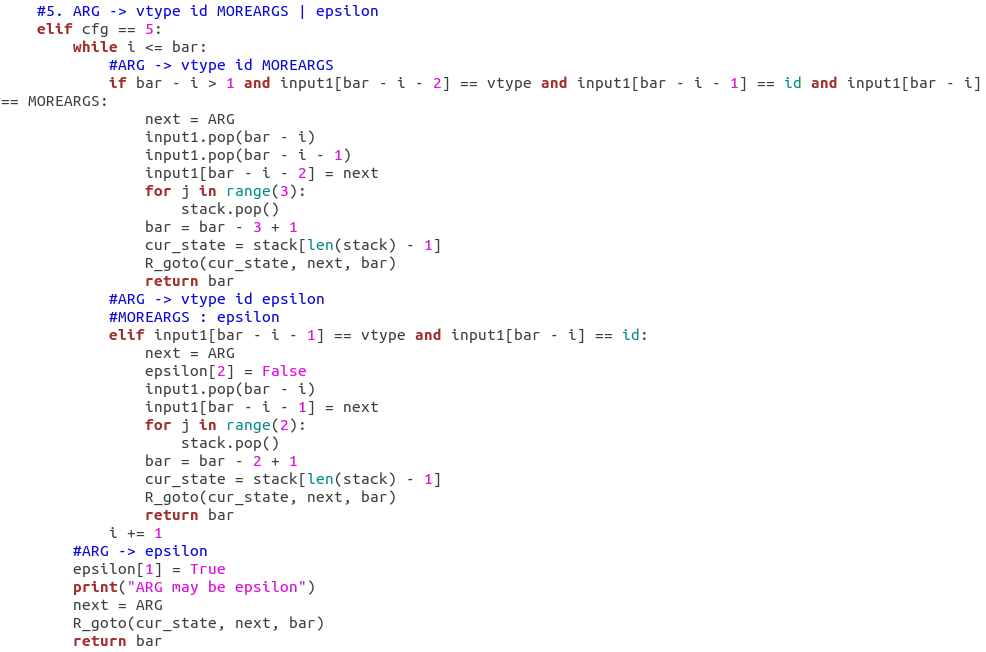
위와 동일하다. bar 이전의 값들(id, assign, RHS)을 찾아 맞는 값이 있으면 reduce, goto를 실행하고 pop된 만큼 bar도 이동한 후에 bar값을 return해준다. 맞는 값이 없을 경우 reject 후 CFG3 ERROR를 출력한다.

04: FDECL → vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace

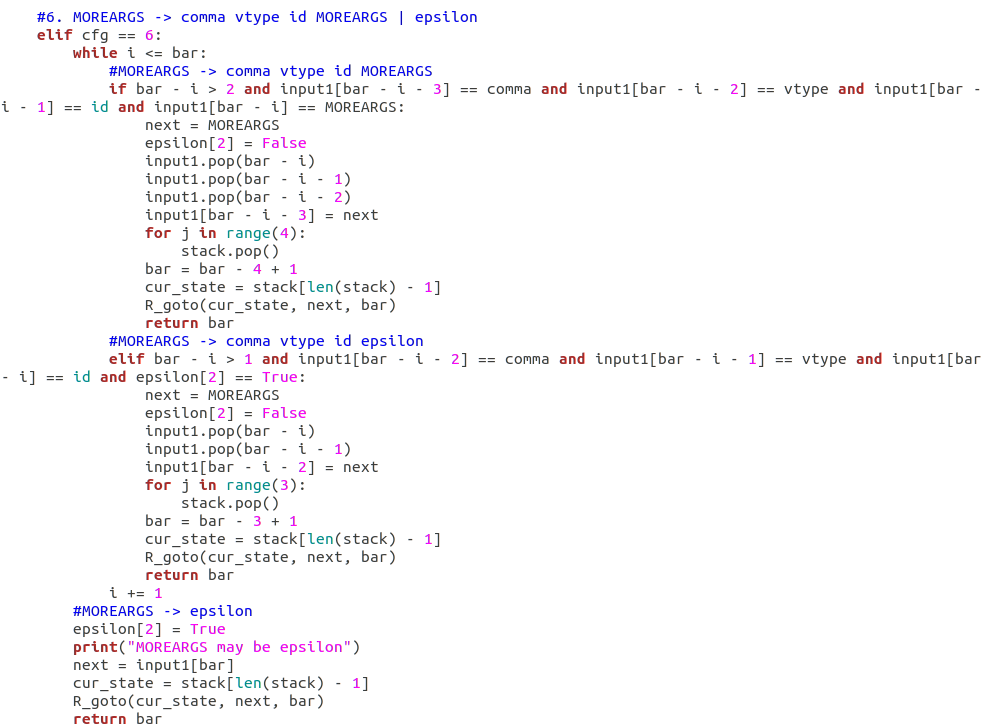
FDECL의 CFG와 맞는지 확인한다 이 때 ARG나 BLOCK이 epsilon이어서 포함되지 않은 경우도 있을 것이기 때문에 그 경우 그 중 하나나 둘이 없다고 생각한 CFG를 시행한다.(CFG의 해당부분만 없는 것으로 생각). BLOCK이 없다면 BLOCK을 제외하고 일치하는 값이 있는지를 보고 ARG가 없다면 ARG를 제외하고 일치하는 값을 확인한다. 그리고 해당 개수에 맞게 pop을 해준 후 FDECL을 넣어준 후 bar를 움직인다.(pop수 – 1 : FDECL넣어주었으니 덜 움직임)pop된 후의 state에 해당하는 곳에 FDECL이 들어왔다면 어떻게 움직일지 R\_goto함수를 실행한다.

만약 일치하는 것이 없다면 reject 후 CFG4 ERROR를 출력한다.



05: ARG → vtype id MOREARGS | ϵ

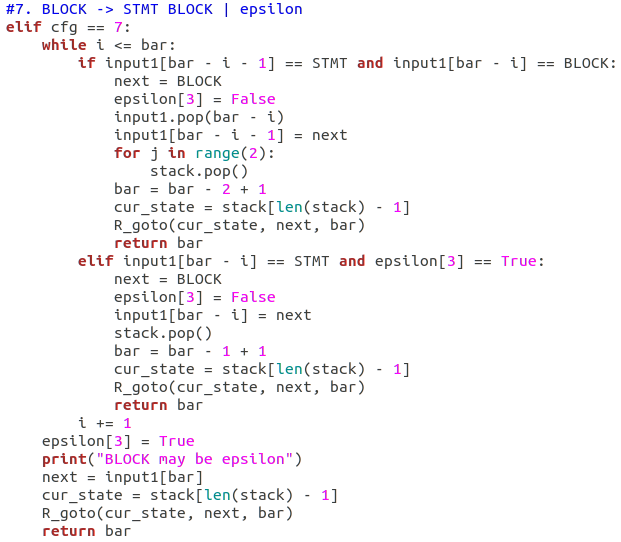
MOREARGS가 epsilon일 경우를 생각한다. 그리고 일치하는 값이 없을 경우 ARG가 epsilon이라고 가정한 후에 R\_goto를 실행한다.(그러나 FOLLOW값에 따라 rparen이 올 경우 ARG 자체를 실행하지 않아 넘어갈 문장이기도 하다. DFA그림에 맞게 넣어주기만 했음) (시연을 위해 epsilon임을 가정하는 부분에 출력을 넣었다. 어떻게 돌아가는지 간단한 예시를 보여준 후 삭제될 문장이다.) 그게 아니라면 MOREARGS까지 포함하여 총 3개의 stack pop, input1 3개 pop 후 bar를 input에 맞춰 이동한다.(이동할 때 pop된후 들어간 것도 생각한다.) ARG를 넣어주고 GOTO를 실행한다. MOREARGS가 epsilon인 경우도 존재하므로 만약 MOREARGS가 epsilon이라면 2개만 pop한다.



06: MOREARGS → comma vtype id MOREARGS | ϵ

뒤에 나오는 MOREARGS가 epsilon일 경우를 생각한다. epsilon이 아니라면 뒤에 나온 MOREARGS까지 포함하여 stack, input1 pop후 MOREARGS를 넣고 bar를 이동시킨다.

만약 일치하는 것이 없다면 앞의 MOREARGS를 epsilon으로 만든다.(MOREARGS epsilon조건을 통과하기 위해서) 그후 R\_goto를 통해 다음 state로 이동한다. (bar가 다음 state에선 MOREARGS의 끝까지 이동한 상태이다)



07: BLOCK → STMT BLOCK | ϵ

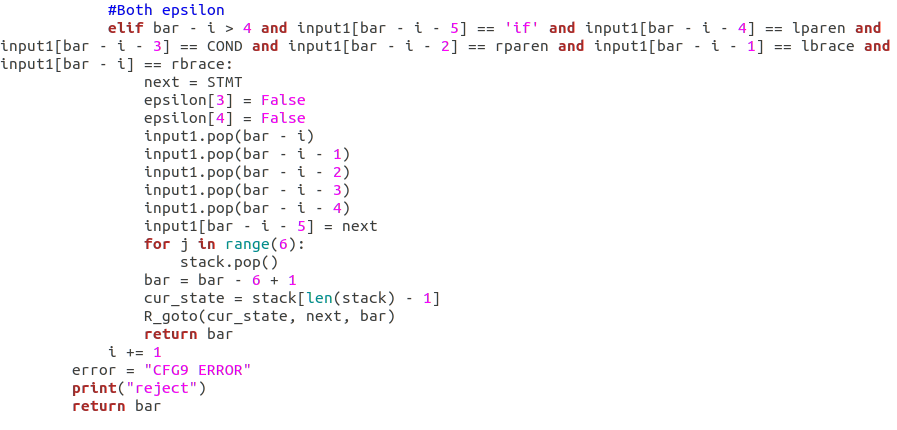
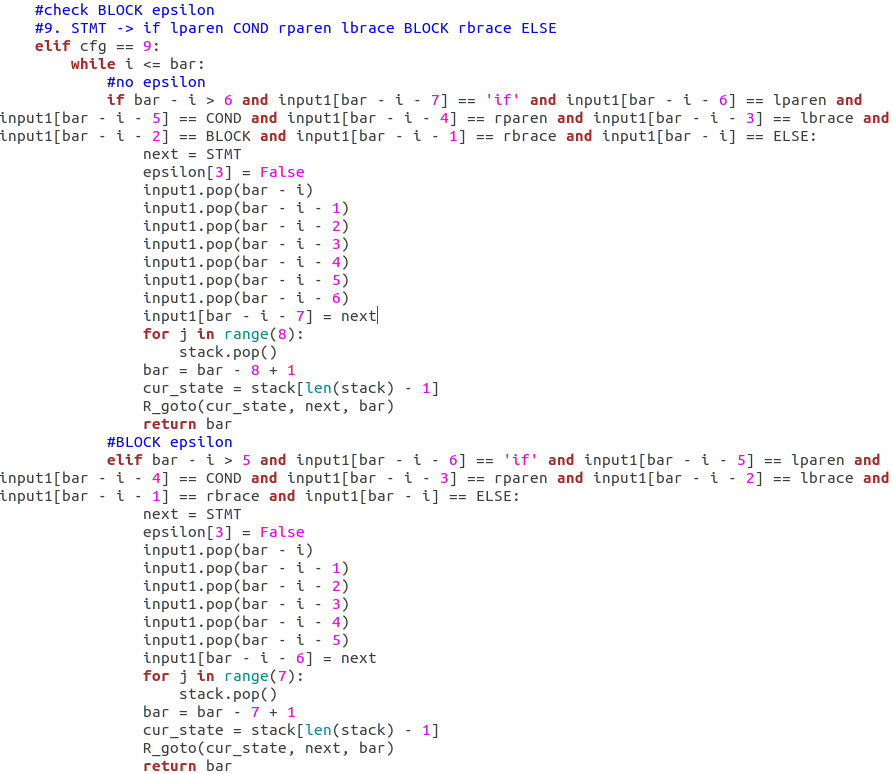
위와 동일하게 BLOCK도 실행한다. 만약 앞서서 BLOCK에 epsilon이 나왔다면 STMT만 있는 상태일 수도 있고, BLOCK이 epsilon인 경우도 있다. 그것에 맞춰서 실행한다. STMT, BLOCK모두 들어있다면 2개 pop후 BLOCK 넣기, STMT만 들어있고 BLOCK가 epsilon이 앞서 나왔다면 STMT를 BLOCk으로, 일치하는 것이 없다면 BLOCK를 epsilon이라고 생각하고 다음 state를 진행한다.



08: STMT → VDECL | ASSIGN semi

STMT가 VDECL 혹은 ASSIGN semi인 경우이다. 둘 모두 epsilon인 경우가 없으니 VDECL의 경우 VDECL을 input1에서 pop하고 STMT를 넣어준다. stack에 들어있는 state도 pop해준 후 goto를 실행한다. bar도 이동한다. ASSIGN semi의 경우도 마찬가지이다. 2개 pop후 STMT하나를 넣어준 후 bar이동, stack도 2개 pop 후 pop된 후의 current state에 맞춰 goto를 실행한다.

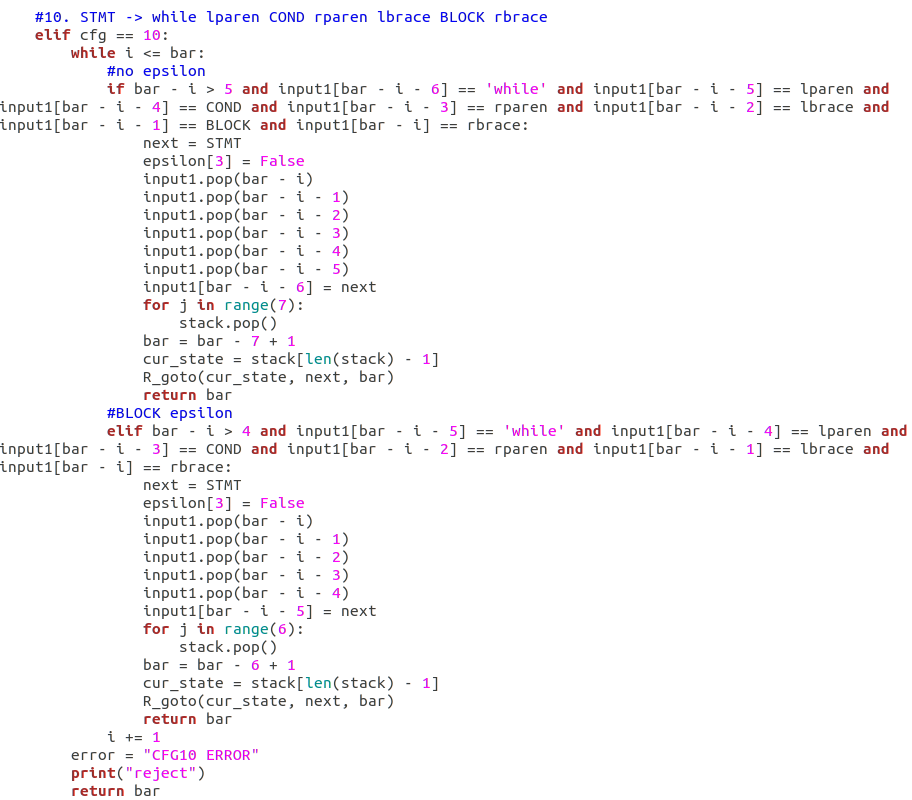
일치하는 것이 없다면 error에 CFG08 ERROR를 넣어 reject한 후 에러를 출력한다.



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

09번의 STMT의 경우, epsilon이 올 수 있는 것이 BLOCK과 ELSE가 있다. BLOCK이 epsilon이 없는 경우 총 8개가 pop될 것이고, BLOCK이 epsilon인 경우, ELSE가 epsilon인 경우는 7개, 둘 모두 epsilon이면 6개가 pop될 것이다. 이것은 BLOCK빼고, ELSE빼고 나머지가 모두 일치하는지를 확인하는 것으로 판단한다. (bar – i – n : bar – i – n이 음수이면 안되므로 넣은 조건) 그리고 각 상황에 맞게 pop한 후 next에 STMT를 넣은 후 STMT가 왔을 때 pop한 후의 state가 어디로 GOTO하는지를 찾는다.

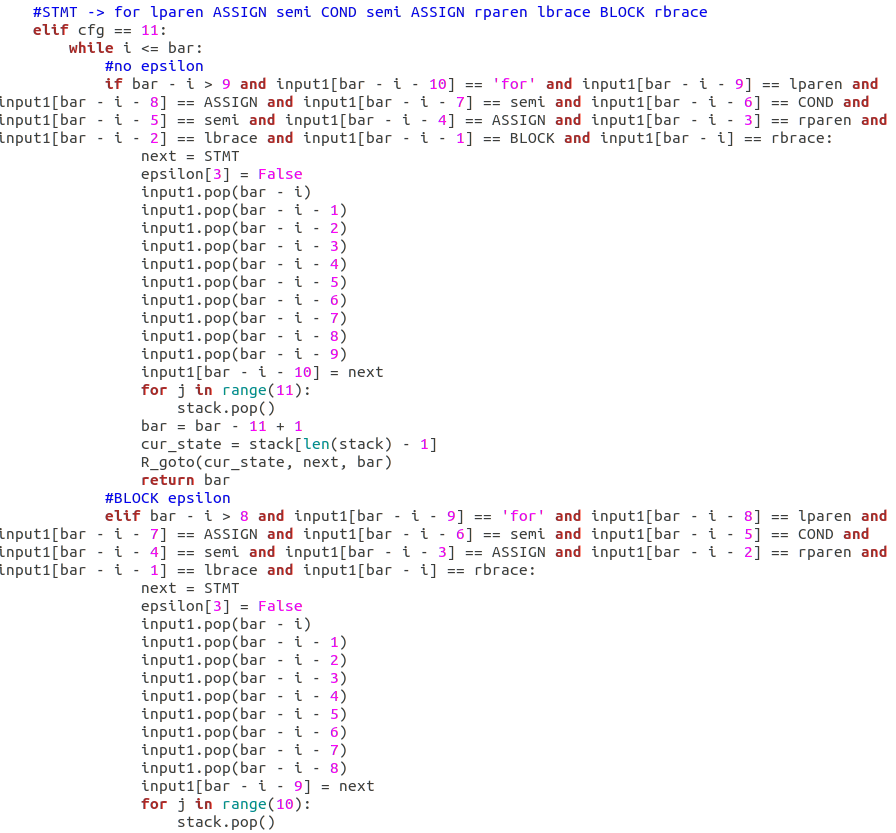
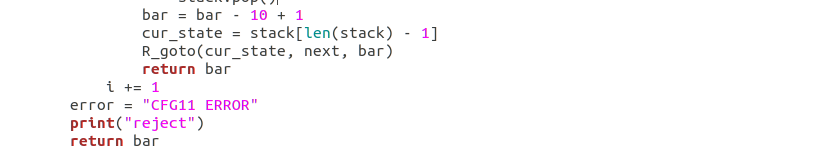
위 STMT는 epsilon인 경우가 없다. 따라서 각 경우에 맞지 않는다면 error에 “CFG9 ERROR”를 넣어 reject한 후 while문을 종료 시킨 후 에러가 발생했음을 알린다.



10: STMT → while lparen COND rparen lbrace BLOCK rbrace

BLOCK만 epsilon일 가능성이 있다. 그 외에는 BLOCK이 epsilon인지 아닌지에 따라 STMT를 찾아보고(while lparen COND rparen lbrace BLOCK rbrace 아니면 while lparen COND rparen lbrace rbrace를 찾는다) 각 상황에 맞게 pop한 후 goto한다.

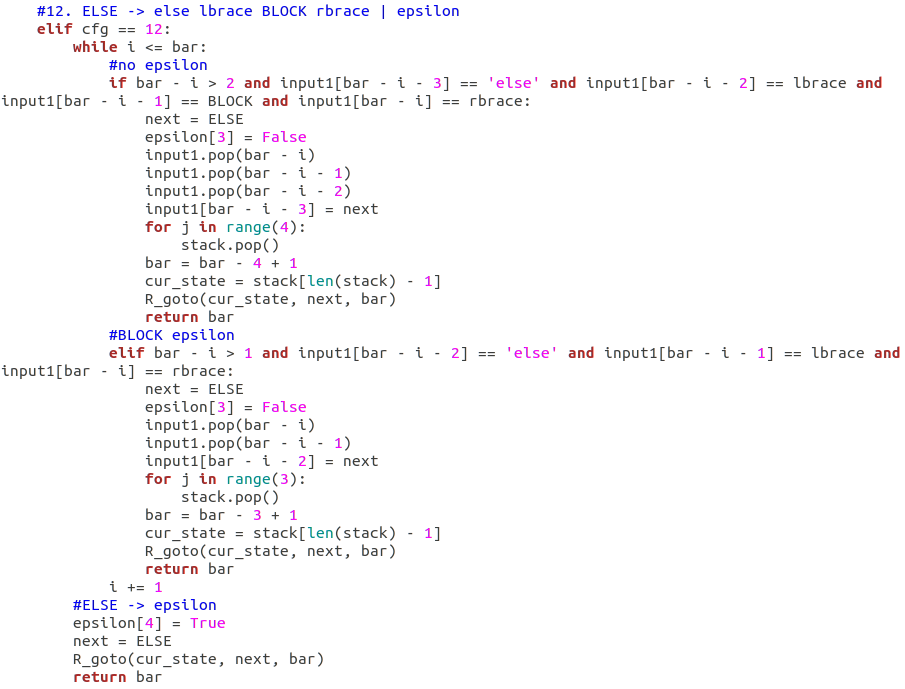
일치하는 것이 없다면 reject한 후 CFG10 ERROR를 출력한다.

11: STMT → for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace

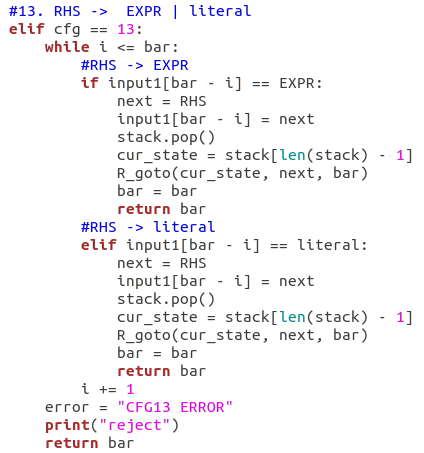
10 Rule과 동일하다. pop할 때 찾는 것만 for lparen ASSIGN semi COND semi ASSIGN rparen lbrace BLOCK rbrace 혹은 BLOCK 없이 찾는 것으로 바뀐다.

일치하는 것이 없다면 reject한 후 CFG11 ERROR를 출력한다.



12: ELSE → else lbrace BLOCK rbrace | ϵ

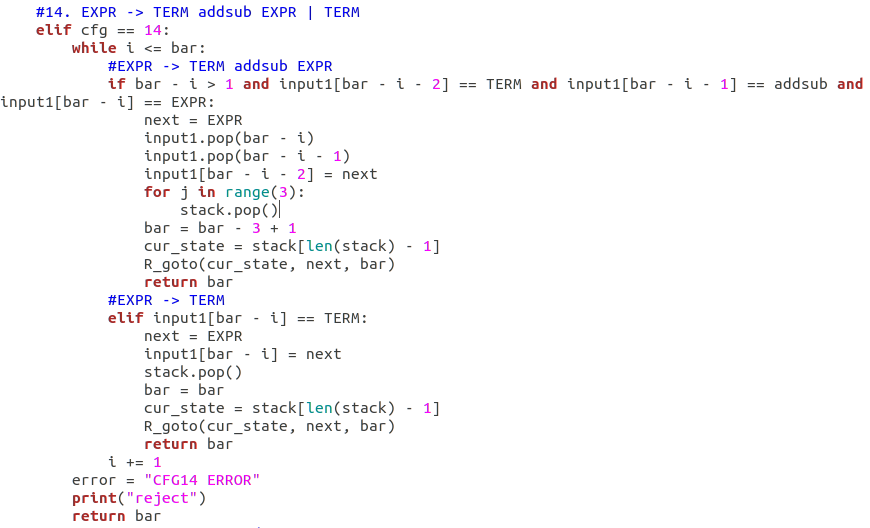
ELSE는 epsilon이 될 수 있다. 따라서 CFG에 나타난 else lbrace BLOCK rbrace 혹은 else lbrace rbrace와 일치하는 것이 없다면 else를 epsilon으로 지정한 후 진행한다. 아니라면 3개 혹은 4개를 stack, input1에서 pop한 후 input1에 ELSE를 넣어주고 bar는 input1을 따라 움직여준다. 그리고 state에 맞는 goto를 실행한다.



13: RHS → EXPR | literal

RHS는 literal(“”안에 들어있는 값)이거나 혹은 다른 CFG RULE을 통해서 나타난 EXPR일 경우 RHS로 바꿔준다. input1은 해당하는 곳을 EXPR, literal에서 RHS로 바꾸어주고, stack은 1개 pop해준후 pop한 곳의 RHS가 가는 곳에 해당하는 곳에goto한다.

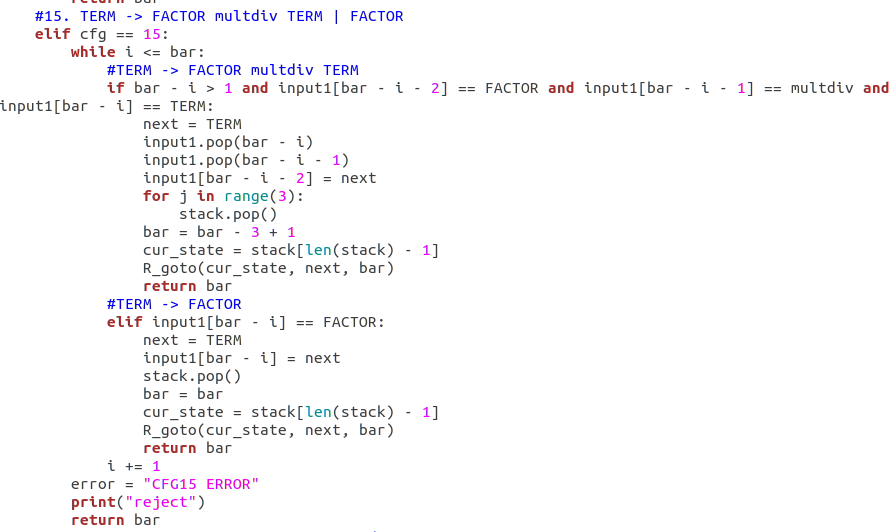
일치하는 것이 없다면 reject를 출력한 후 CFG13 ERROR를 출력한다.



14: EXPR → TERM addsub EXPR | TERM

EXPR은 TERM addsub EXPR가 bar의 앞쪽에 있는 경우 3개 pop해주고, input1에 EXPR을 넣어주고 pop한 state에 해당하는 값에서 EXPR이 나왔을 때 goto를 해준다. 그 후bar를 움직인다. 혹은 TERM이 나왔을 경우 EXPR로 바꾸어주고 stack하나 pop해준 후 해당 state에서 EXPR이 나왔을 경우 goto하는 것을 실행한다. 일치하는 것이 없다면 error는 CFG14 ERROR를 출력한다.

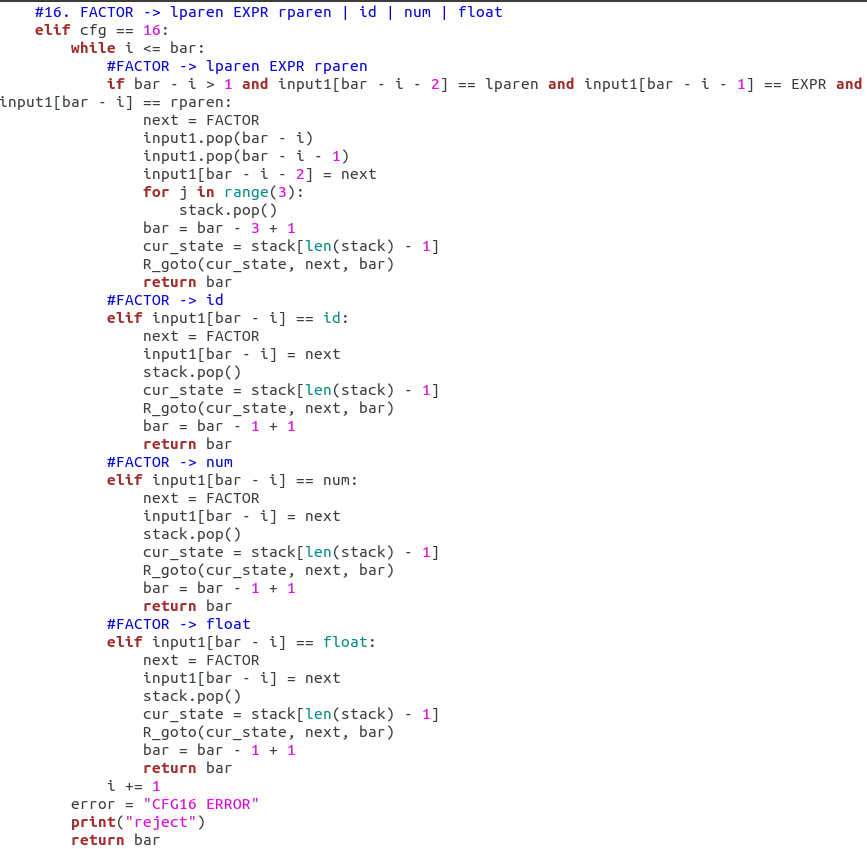
이 EXPR문을 통해 무한 +, -가 가능해진다.



15: TERM → FACTOR multdiv TERM | FACTOR

TERM은 위의 15의 EXPR과 비슷하다. bar 앞쪽에 해당하는 값이 있다면 알맞게 pop해준 후 goto해준다. bar도 움직인다. 이 때 일치하는 것이 없다면 reject 후 error는 CFG15 ERROR를 출력한다.

이 CFG를 통해 무한 \* , /이 가능해진다.



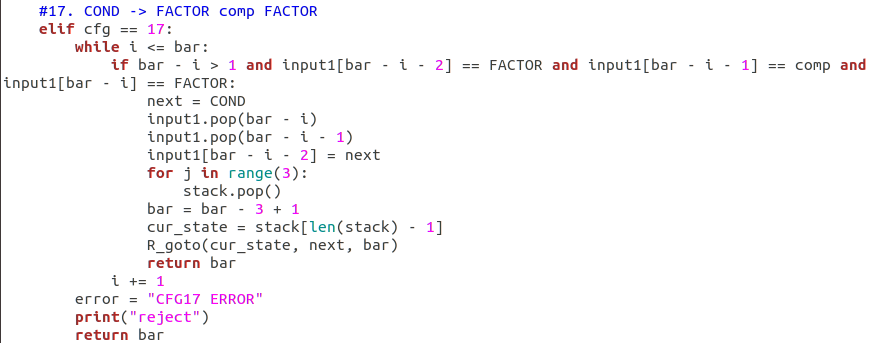
16: FACTOR → lparen EXPR rparen | id | num | float

FACTOR는 양옆이 괄호로 감싸인 EXPR이나 id, num, float의 경우가 있다. 만약 lparen, EXPR, rparen이라면 Rule에 따라 input1을 3개를 pop하고, state도 마찬가지로 3개를 pop한 후 FACTOR를 input1에 넣어준다. 그리고 bar를 움직인다. stack은 현재 pop된 후 state에 따라 FACTOR 나왔을 때의 goto를 실행한다.

그 외에 id, num, float의 경우 한 개씩 pop해준 후 해당 id, num, float에 맞는 값을 FACTOR로 바꿔주고 goto해준다.

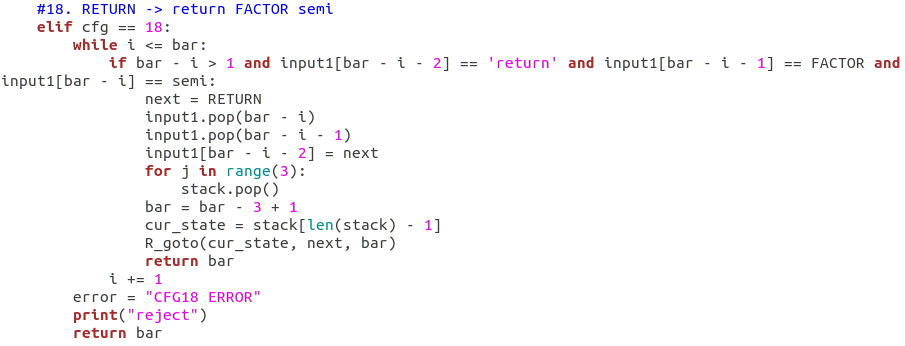
만약 일치하는 것이 하나도 없다면 reject후 error CFG16 ERROR를 출력한다.

이 CFG를 통해 무한 괄호 감싸기가 가능해진다.



17: COND → FACTOR comp FACTOR

어디에도 나올 수 있는 epsilon없다. FACTOR comp FACTOR가 순서대로 나온 것을 bar앞쪽에서 찾은 후 일치하는 것이 있다면 3개 pop, COND를 넣고 bar움직이고 현 state에 COND나왔을 때 goto를 실행한다. 만약 일치하는 것이 없다면 reject, CFG17 ERROR를 출력한다.



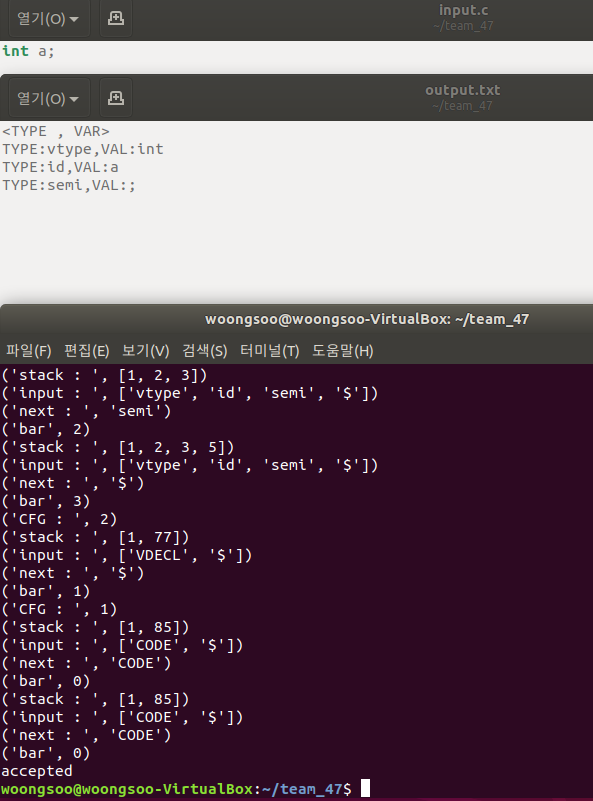
18: RETURN → return FACTOR semi

RETURN도 epsilon이 나올 수 있는 요소가 없다. 따라서 bar 앞쪽에 return FACTOR semi가 나온다면 해당 요소들(3개)를 stack, inpu1에서 pop해준 후 RETURN을 input1에 넣고 bar를 그만큼 움직인다.(2개) 그리고 pop된 후의 현재 state에 RETURN이 들어왔을 때인 goto를 실행한다.

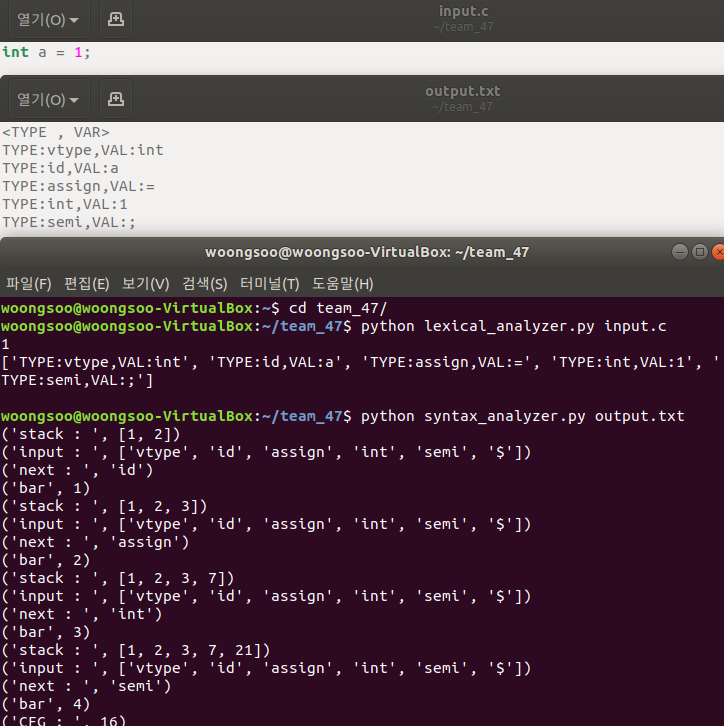
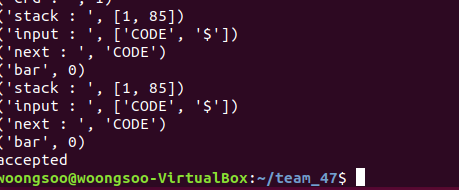
만약 일치하는 것이 나오지 않았다면 reject후, CFG18 ERROR를 띄운다.

6. RUN CODE

CFG에 맞는 code를 input에 예시로 넣어보면서 돌려본다.

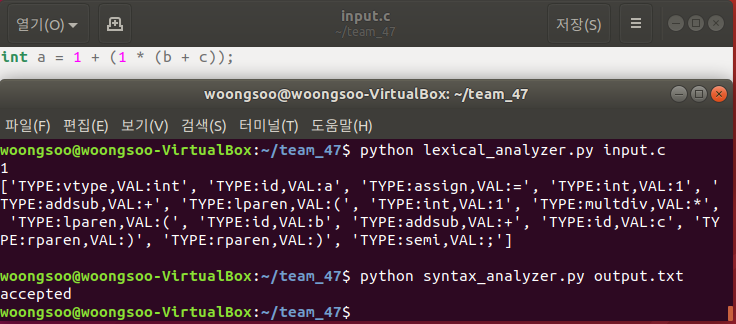


2번인 VDECL에 맞는 코드를 돌려보면 위와 같은 식으로 stack이 쌓이고, bar가 이동하고, CFG2 RULE에 따라 VDECL로 REDUCE되면서 CODE $로 input이 정리된 후에 accepted가 되는 것을 확인할 수 있다.

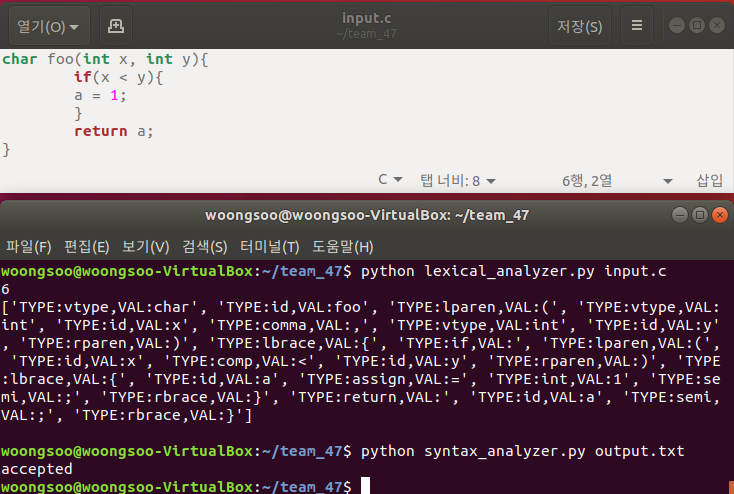
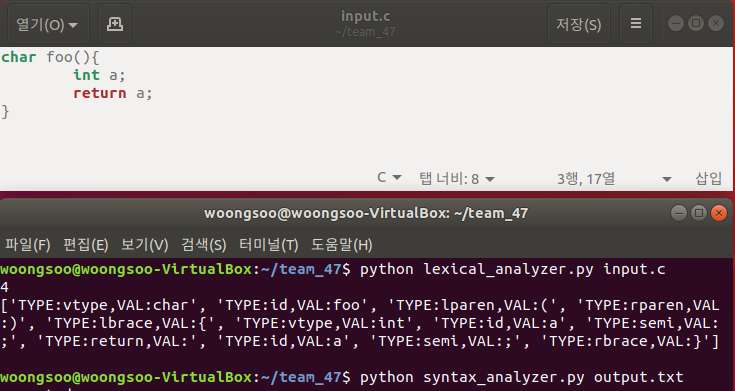
 

또 VDECL -> id ASSIGN semi의 경우도 위와같이 state들을 ASSIGN에 맞는 CODE를 찾기 위해 FACTOR, TERM, EXPR, RHS등의 이동도 되며 reduce가 이루어지는 모습을 확인할 수 있다.

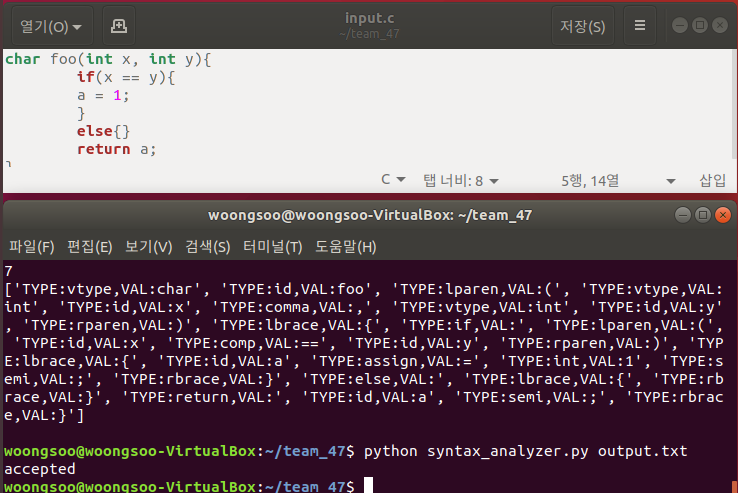
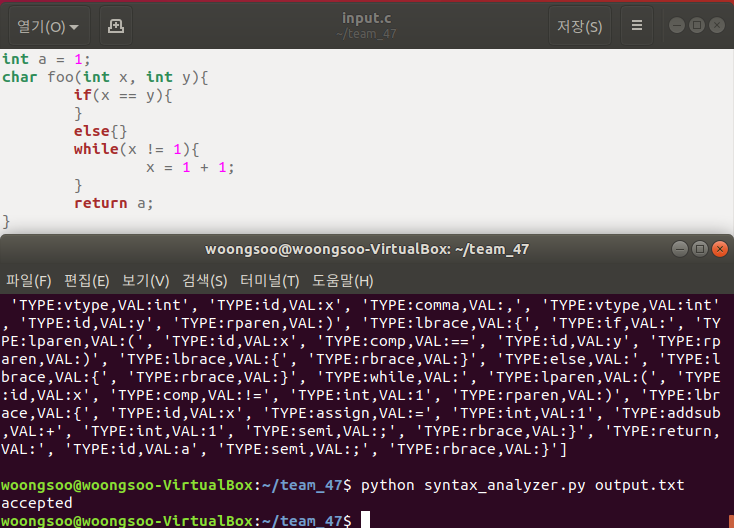
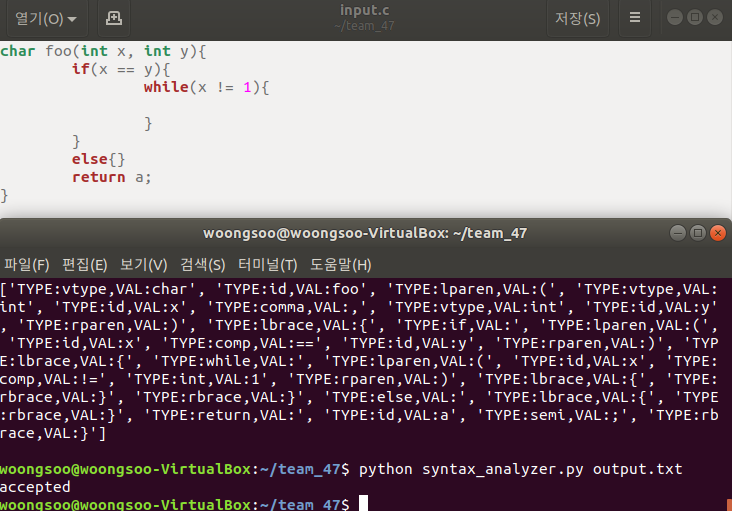
이제 어떤 식으로 작동하는지는 충분히 확인한 것 같으니 accept, reject메시지만을 띄워 각각의 CFG에 맞게 실행해보도록 한다.



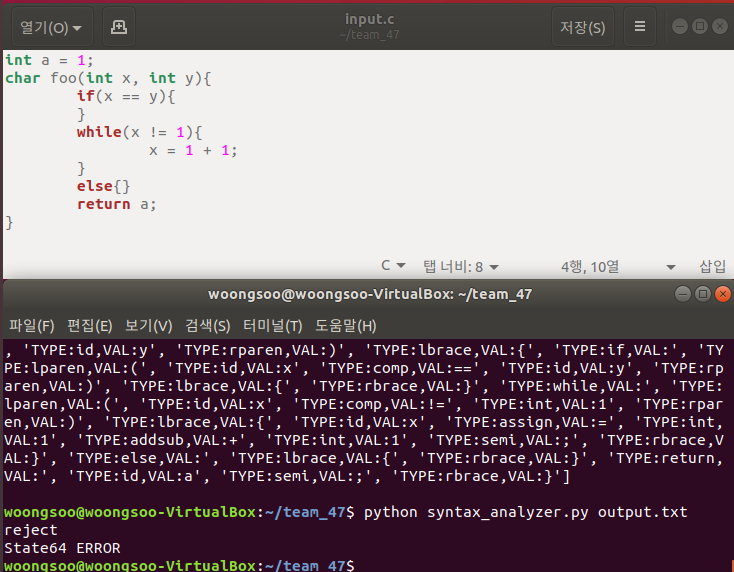
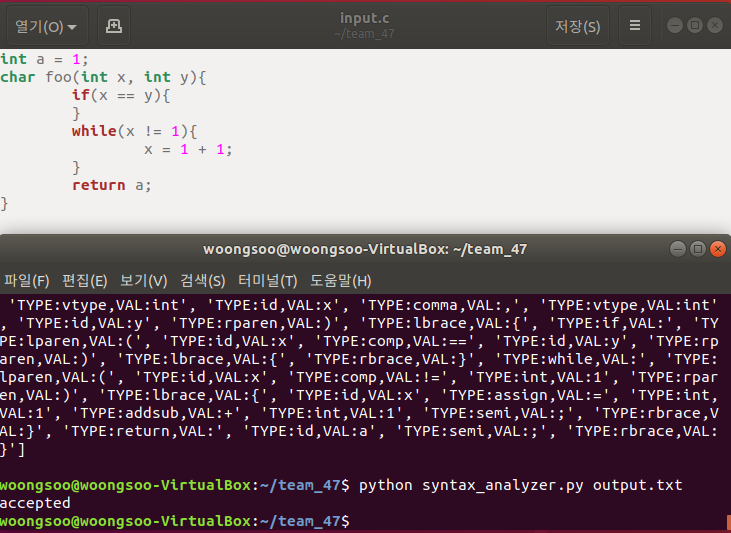
literal 중첩 paren, + \*등 가지가지 들어가는지



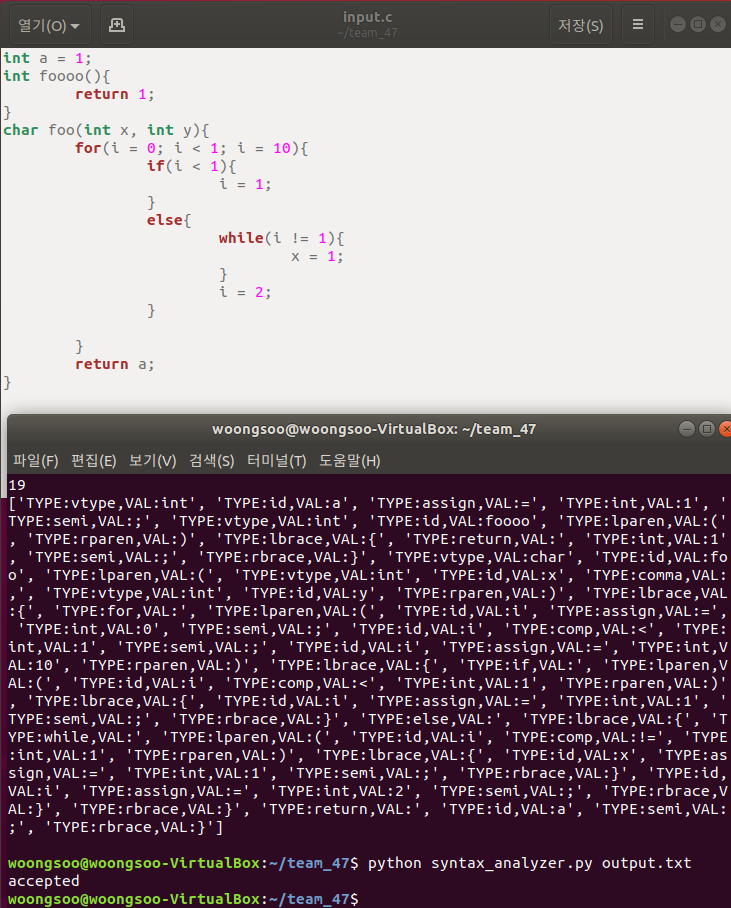
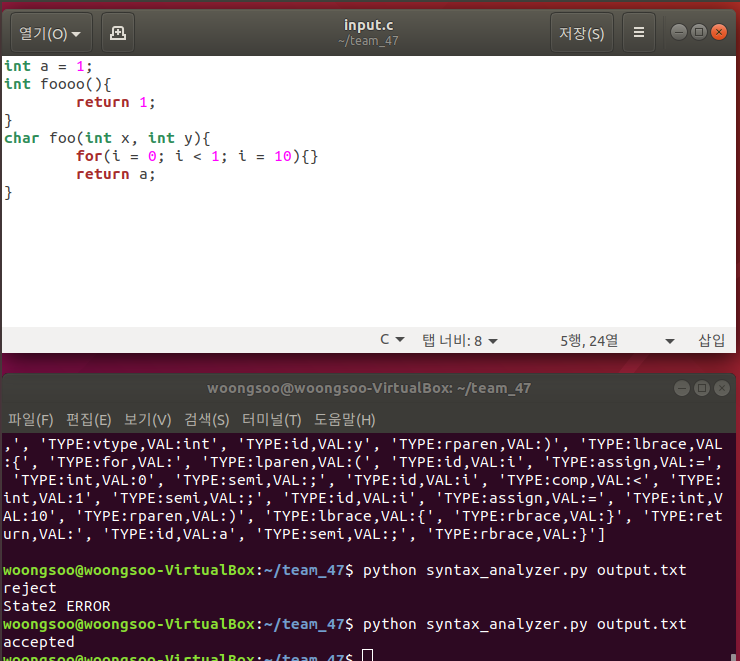
ARG빈채로 실행되는지 if문

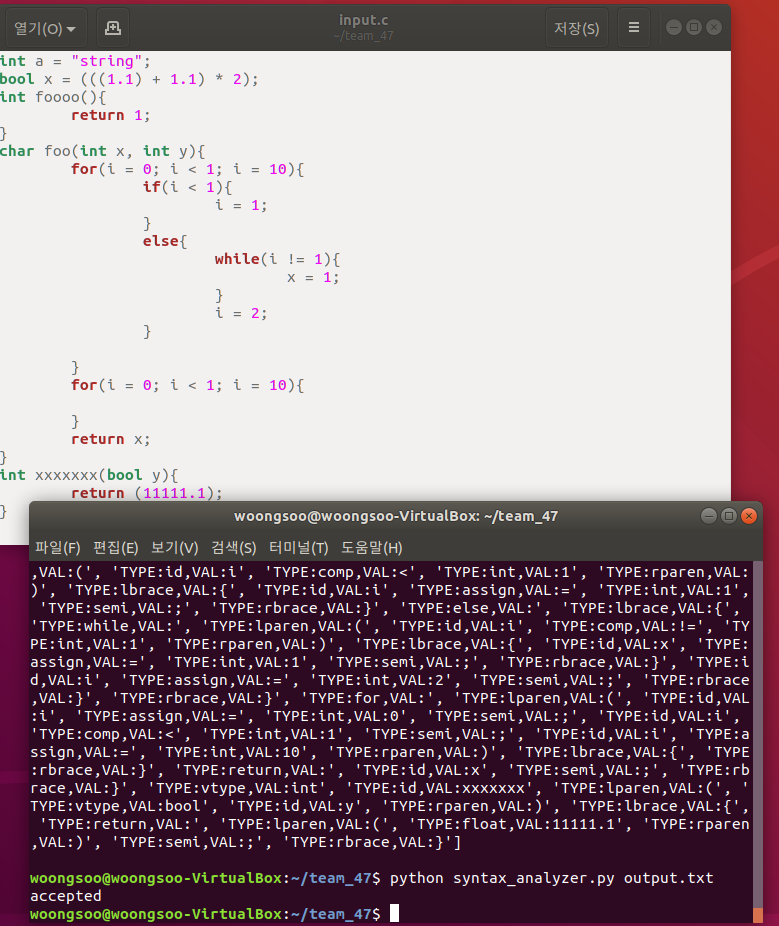
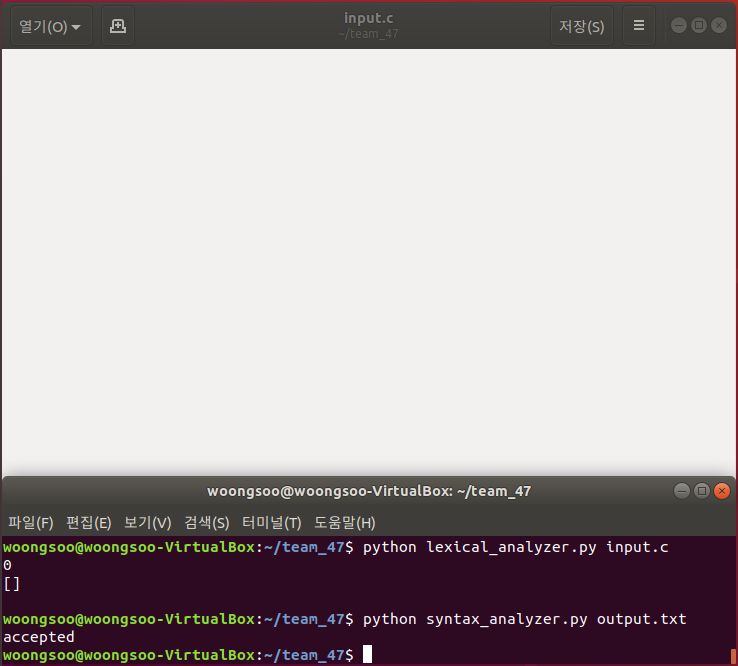
ifelse if문 내부에 while문 if문 외부에 while문

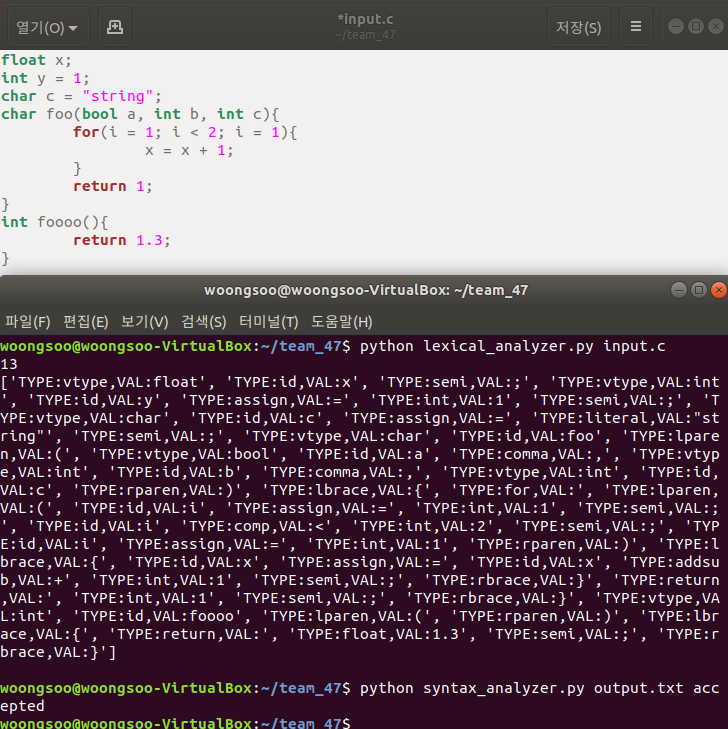
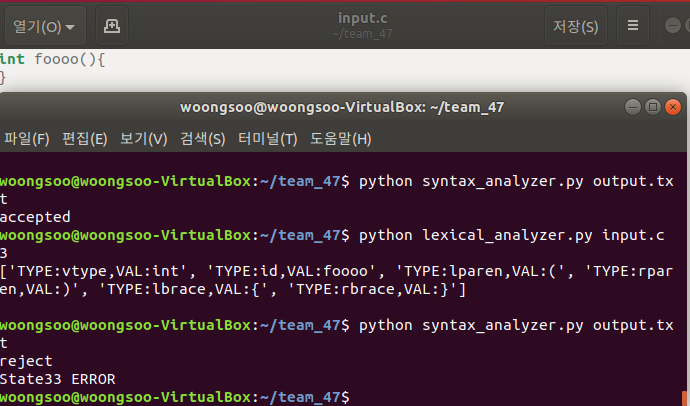
만약 while문과 else의 자리가 바뀐다면 : error / else가 안나온다면 not error



for문 및 여러 함수와 변수(for문 마지막 arguments가 ASSIGN이다) / 중첩의 중첩의 중첩함수



input값이 비어있는 경우 여러 가지 경우



return을 넣지 않는다. MOREARGS넣은 경우