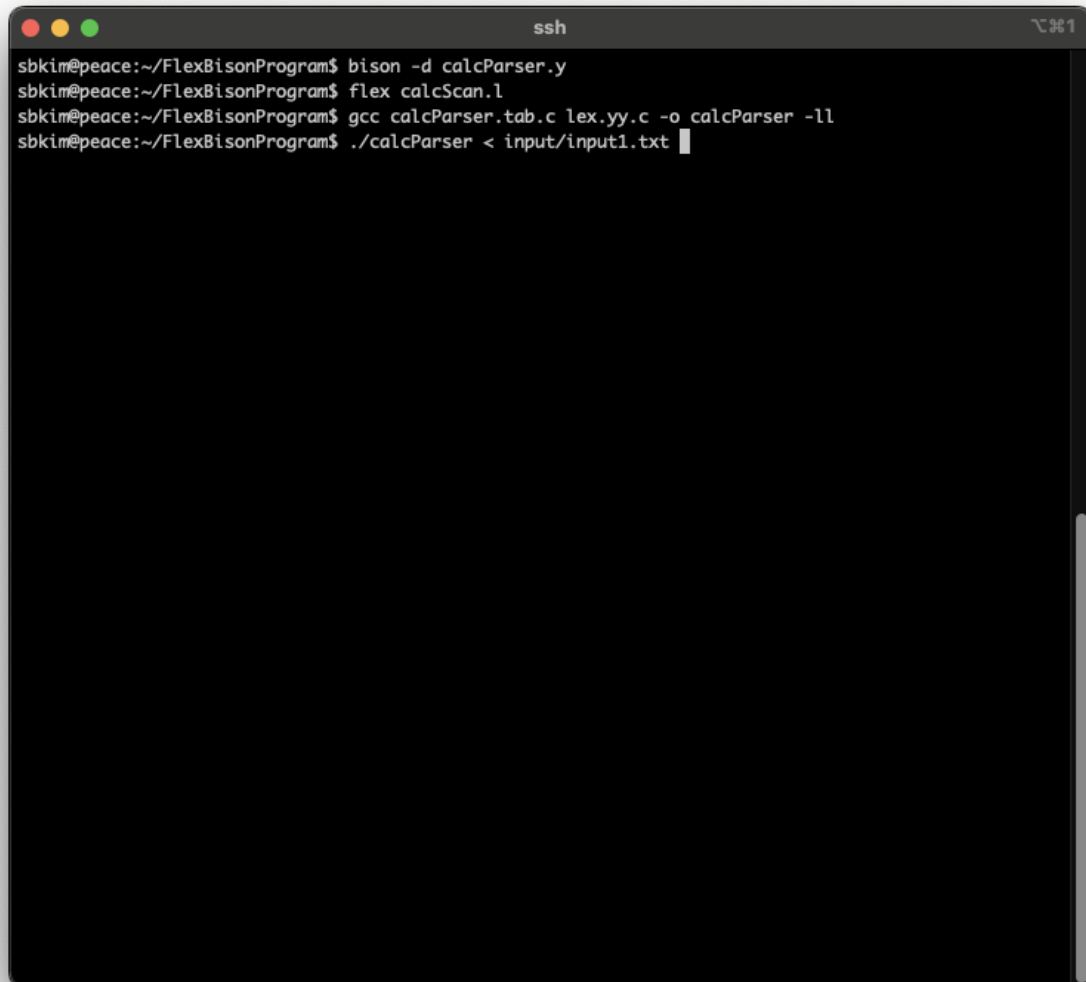


22100113 Seongbin Kim
Compiler Theory
June 24, 2024
Flex Bison Program

Command Sequence and Generated Output

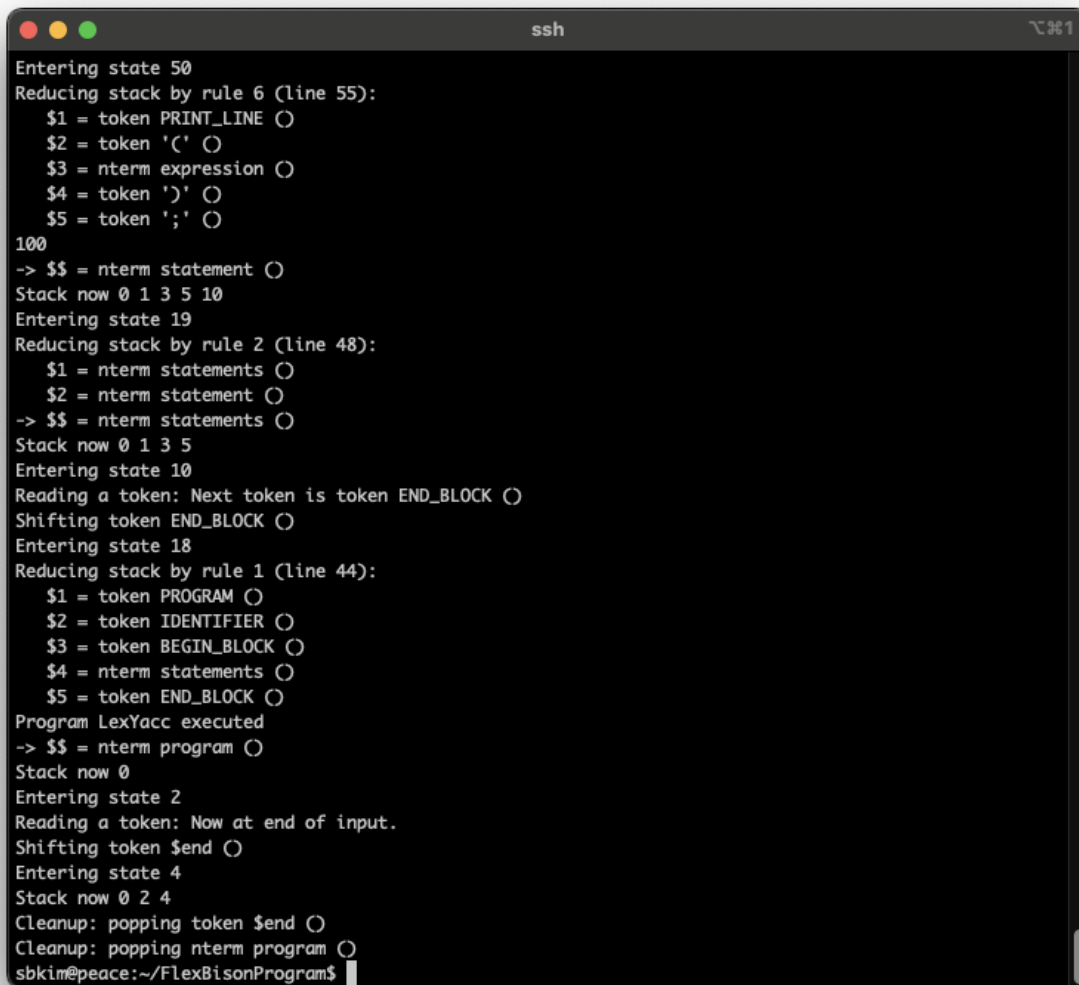
Build command sequence



```
ssh
sbkim@peace:~/FlexBisonProgram$ bison -d calcParser.y
sbkim@peace:~/FlexBisonProgram$ flex calcScan.l
sbkim@peace:~/FlexBisonProgram$ gcc calcParser.tab.c lex.yy.c -o calcParser -ll
sbkim@peace:~/FlexBisonProgram$ ./calcParser < input/input1.txt
```

Running Output

input/input1.txt

A terminal window titled 'ssh' with a dark background and light gray text. The window shows the execution of a LexYacc program. The output includes state transitions, stack operations, and token shifts. The stack starts empty and grows as tokens are shifted. The program eventually reaches the end of the input and performs cleanup by popping tokens and nterms from the stack. The prompt 'sbkim@peace:~/FlexBisonProgram\$' is visible at the bottom.

```
Entering state 50
Reducing stack by rule 6 (line 55):
  $1 = token PRINT_LINE ()
  $2 = token '(' ()
  $3 = nterm expression ()
  $4 = token ')' ()
  $5 = token ';' ()
100
-> $$ = nterm statement ()
Stack now 0 1 3 5 10
Entering state 19
Reducing stack by rule 2 (line 48):
  $1 = nterm statements ()
  $2 = nterm statement ()
-> $$ = nterm statements ()
Stack now 0 1 3 5
Entering state 10
Reading a token: Next token is token END_BLOCK ()
Shifting token END_BLOCK ()
Entering state 18
Reducing stack by rule 1 (line 44):
  $1 = token PROGRAM ()
  $2 = token IDENTIFIER ()
  $3 = token BEGIN_BLOCK ()
  $4 = nterm statements ()
  $5 = token END_BLOCK ()
Program LexYacc executed
-> $$ = nterm program ()
Stack now 0
Entering state 2
Reading a token: Now at end of input.
Shifting token $end ()
Entering state 4
Stack now 0 2 4
Cleanup: popping token $end ()
Cleanup: popping nterm program ()
sbkim@peace:~/FlexBisonProgram$
```

input/input2.txt

```
ssh 1
$4 = token ')' ()
$5 = token BEGIN_BLOCK ()
$6 = nterm statements ()
$7 = token END_BLOCK ()
$8 = token ELSE ()
$9 = token BEGIN_BLOCK ()
$10 = nterm statements ()
$11 = token END_BLOCK ()
-> $$ = nterm statement ()
Stack now 0 1 3 5 10
Entering state 19
Reducing stack by rule 2 (line 48):
  $1 = nterm statements ()
  $2 = nterm statement ()
-> $$ = nterm statements ()
Stack now 0 1 3 5
Entering state 10
Reading a token: Next token is token END_BLOCK ()
Shifting token END_BLOCK ()
Entering state 18
Reducing stack by rule 1 (line 44):
  $1 = token PROGRAM ()
  $2 = token IDENTIFIER ()
  $3 = token BEGIN_BLOCK ()
  $4 = nterm statements ()
  $5 = token END_BLOCK ()
Program LexYacc2 executed
-> $$ = nterm program ()
Stack now 0
Entering state 2
Reading a token: Now at end of input.
Shifting token $end ()
Entering state 4
Stack now 0 2 4
Cleanup: popping token $end ()
Cleanup: popping nterm program ()
sbkim@peace:~/FlexBisonProgram$
```

input/input3.txt

```
ssh
Reducing stack by rule 3 (line 49):
    $1 = nterm statement ()
-> $$ = nterm statements ()
Stack now 0 1 3 5 10 9 17 32 42 52
Entering state 54
Reading a token: Next token is token IDENTIFIER ()
Shifting token IDENTIFIER ()
Entering state 6
Reading a token: Next token is token '=' ()
Shifting token '=' ()
Entering state 13
Reading a token: Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 21
Reducing stack by rule 20 (line 90):
    $1 = token NUMBER ()
-> $$ = nterm factor ()
Stack now 0 1 3 5 10 9 17 32 42 52 54 6 13
Entering state 26
Reducing stack by rule 19 (line 86):
    $1 = nterm factor ()
-> $$ = nterm term ()
Stack now 0 1 3 5 10 9 17 32 42 52 54 6 13
Entering state 25
Reading a token: Next token is token ';' ()
Reducing stack by rule 16 (line 80):
    $1 = nterm term ()
-> $$ = nterm expression ()
Stack now 0 1 3 5 10 9 17 32 42 52 54 6 13
Entering state 24
Next token is token ';' ()
Reducing stack by rule 13 (line 68):
    $1 = token IDENTIFIER ()
    $2 = token '=' ()
    $3 = nterm expression ()
Error: Undefined variable
sbkim@peace:~/FlexBisonProgram$
```

*All deliverables and source code files are in ~/sbkim/FlexBisonProgram/ for validation.