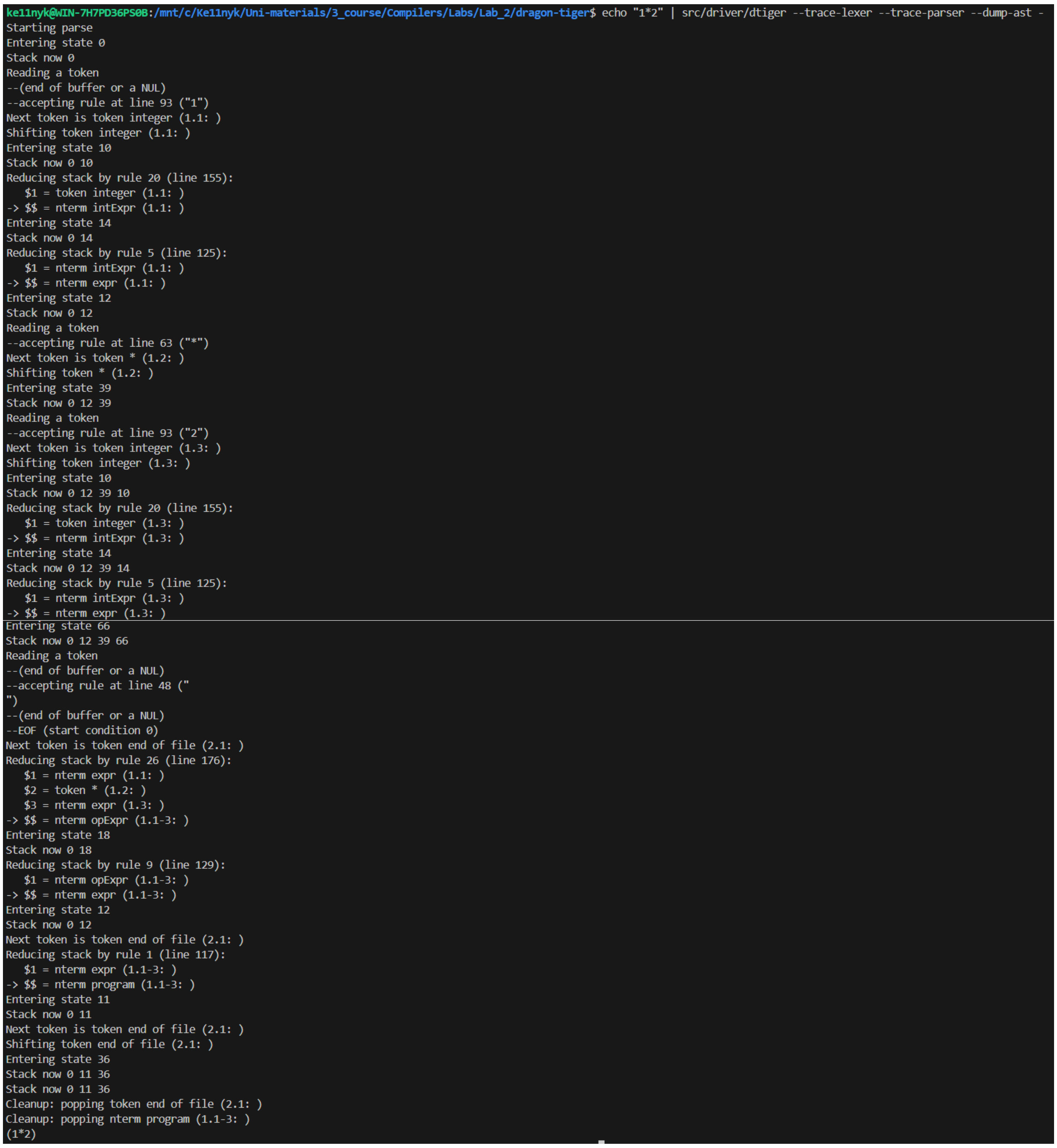
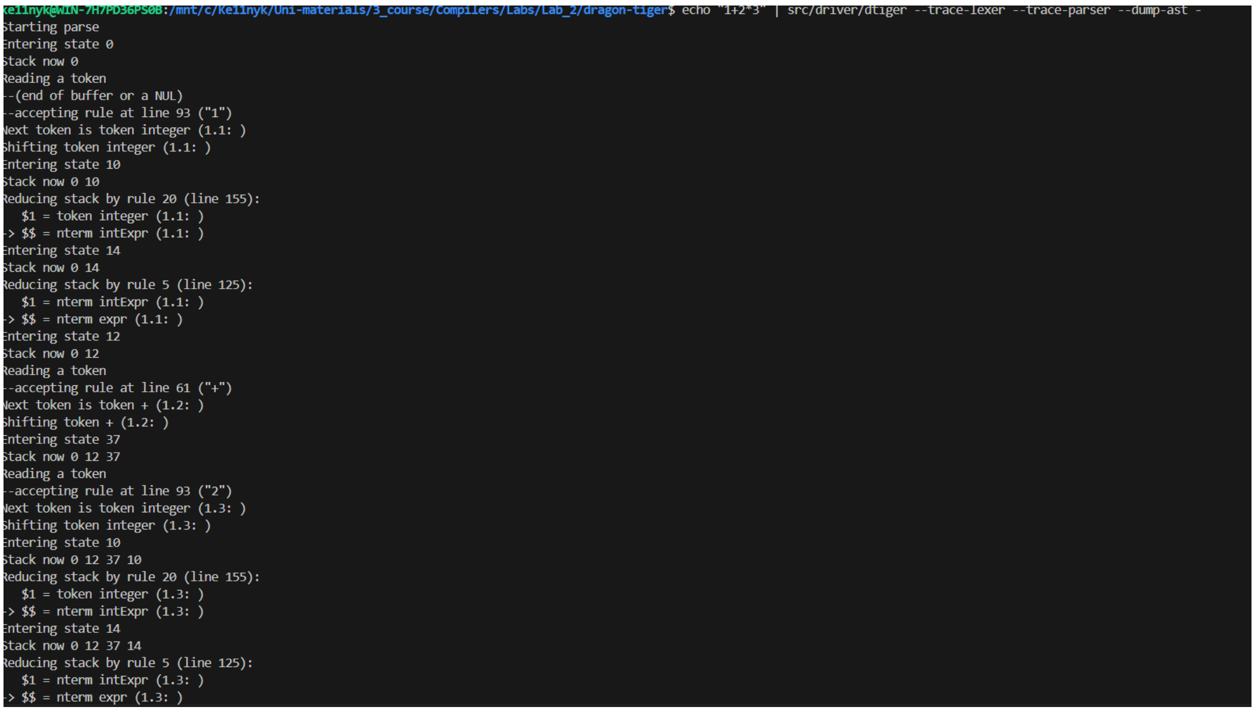
Supporting integers

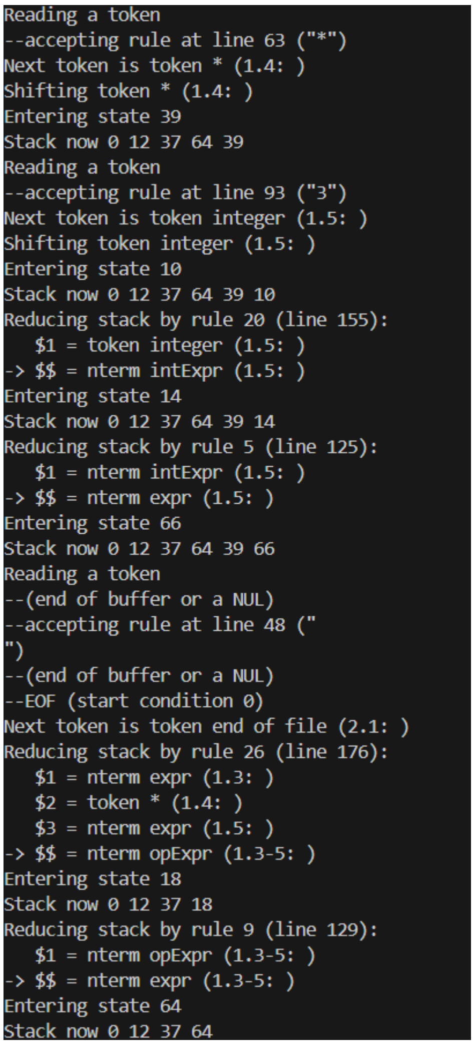


```
ke11nyk@WIN-7H7PD36PS0B:/mnt/c/Ke11nyk/Uni-materials/3_course/Compilers/Labs/Lab_2/dragon-tiger$ echo "1*2" | src/driver/dtiger --trace-lexer --trace-parser --dump-ast -
Starting parse
Entering state 0
Stack now 0
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 93 ("1")
Next token is token integer (1.1: )
Shifting token integer (1.1: )
Entering state 10
Stack now 0 10
Reducing stack by rule 20 (line 155):
   $1 = token integer (1.1: )
-> $$ = nterm intExpr (1.1: )
Entering state 14
Stack now 0 14
Reducing stack by rule 5 (line 125):
   $1 = nterm intExpr (1.1: )
-> $$ = nterm expr (1.1: )
Entering state 12
Stack now 0 12
Reading a token
--accepting rule at line 63 ("*")
Next token is token * (1.2: )
Shifting token * (1.2: )
Entering state 39
Stack now 0 12 39
Reading a token
--accepting rule at line 93 ("2")
Next token is token integer (1.3: )
Shifting token integer (1.3: )
Entering state 10
Stack now 0 12 39 10
Reducing stack by rule 20 (line 155):
   $1 = token integer (1.3: )
-> $$ = nterm intExpr (1.3: )
Entering state 14
Stack now 0 12 39 14
Reducing stack by rule 5 (line 125):
   $1 = nterm intExpr (1.3: )
-> $$ = nterm expr (1.3: )
Entering state 66
Stack now 0 12 39 66
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 48 ("
")
--(end of buffer or a NUL)
--EOF (start condition 0)
Next token is token end of file (2.1: )
Reducing stack by rule 26 (line 176):
   $1 = nterm expr (1.1: )
   $2 = token * (1.2: )
   $3 = nterm expr (1.3: )
-> $$ = nterm opExpr (1.1-3: )
Entering state 18
Stack now 0 18
Reducing stack by rule 9 (line 129):
   $1 = nterm opExpr (1.1-3: )
-> $$ = nterm expr (1.1-3: )
Entering state 12
Stack now 0 12
Next token is token end of file (2.1: )
Reducing stack by rule 1 (line 117):
   $1 = nterm expr (1.1-3: )
-> $$ = nterm program (1.1-3: )
Entering state 11
Stack now 0 11
Next token is token end of file (2.1: )
Shifting token end of file (2.1: )
Entering state 36
Stack now 0 11 36
Stack now 0 11 36
Cleanup: popping token end of file (2.1: )
Cleanup: popping nterm program (1.1-3: )
(1*2)
```

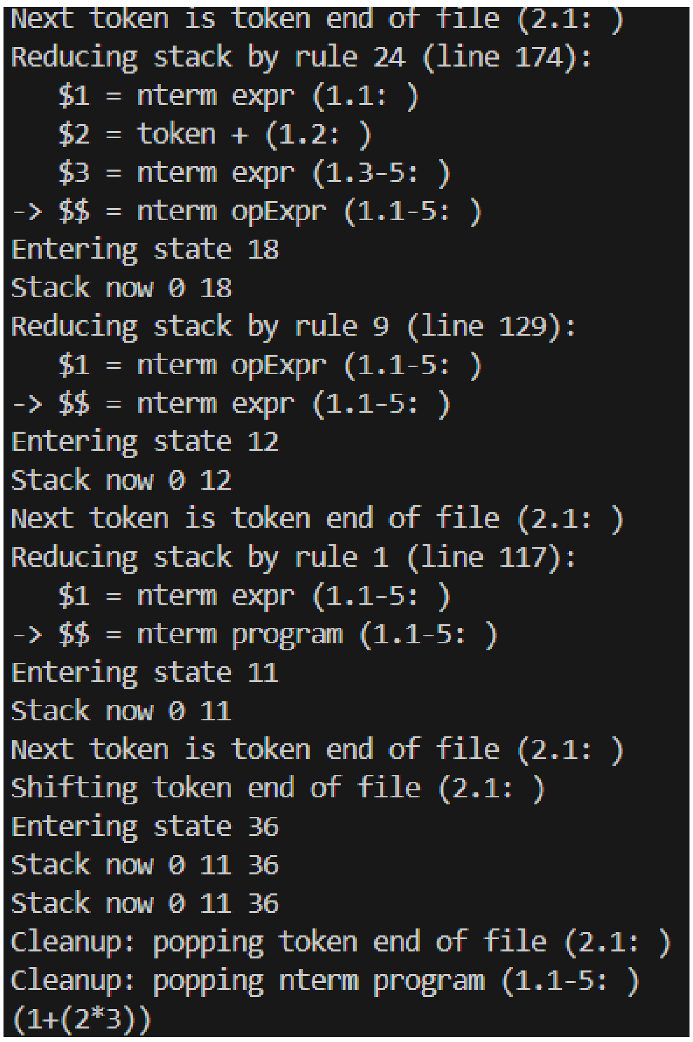
Adding support for more binary operators and adding precedence rules



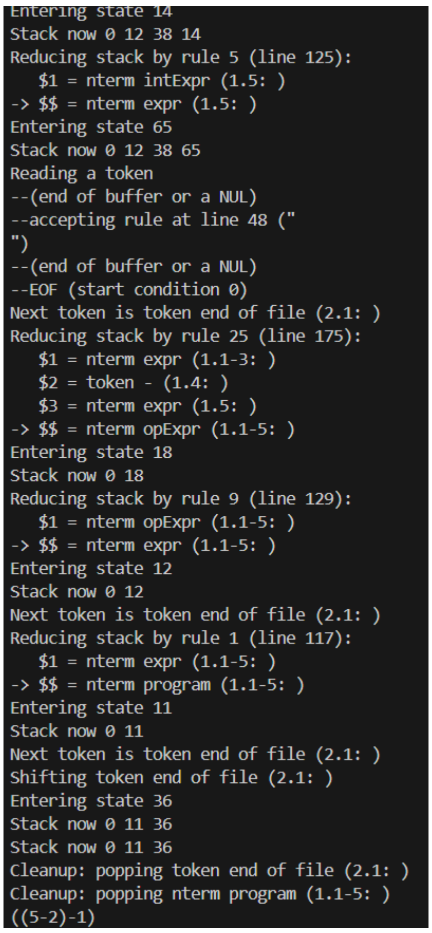
```
ke11nyk@WIN-7H7PD36PS0B:/mnt/c/Ke11nyk/Uni-materials/3_course/Compilers/Labs/Lab_2/dragon-tiger$ echo "1+2*3" | src/driver/dtiger --trace-lexer --trace-parser --dump-ast -
Starting parse
Entering state 0
Stack now 0
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 93 ("1")
Next token is token integer (1.1: )
Shifting token integer (1.1: )
Entering state 10
Stack now 0 10
Reducing stack by rule 20 (line 155):
   $1 = token integer (1.1: )
-> $$ = nterm intExpr (1.1: )
Entering state 14
Stack now 0 14
Reducing stack by rule 5 (line 125):
   $1 = nterm intExpr (1.1: )
-> $$ = nterm expr (1.1: )
Entering state 12
Stack now 0 12
Reading a token
--accepting rule at line 61 ("+")
Next token is token + (1.2: )
Shifting token + (1.2: )
Entering state 37
Stack now 0 12 37
Reading a token
--accepting rule at line 93 ("2")
Next token is token integer (1.3: )
Shifting token integer (1.3: )
Entering state 10
Stack now 0 12 37 10
Reducing stack by rule 20 (line 155):
   $1 = token integer (1.3: )
-> $$ = nterm intExpr (1.3: )
Entering state 14
Stack now 0 12 37 14
Reducing stack by rule 5 (line 125):
   $1 = nterm intExpr (1.3: )
-> $$ = nterm expr (1.3: )
```



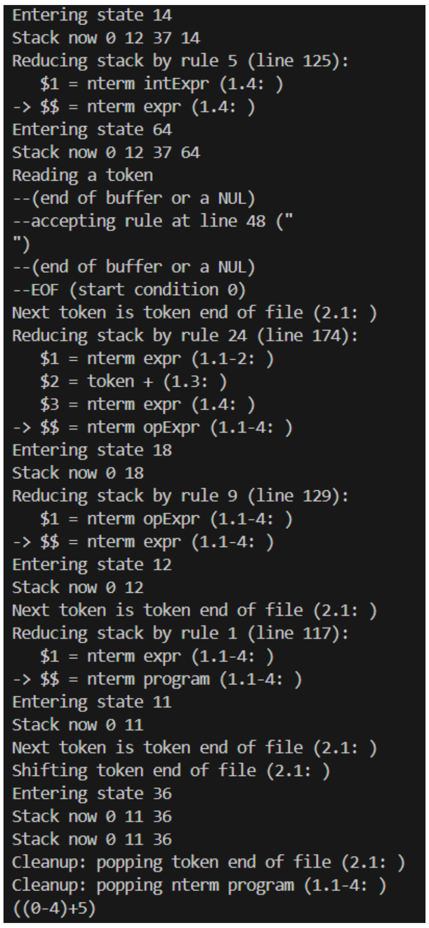
```
Reading a token
--accepting rule at line 63 ("*")
Next token is token * (1.4: )
Shifting token * (1.4: )
Entering state 39
Stack now 0 12 37 64 39
Reading a token
--accepting rule at line 93 ("3")
Next token is token integer (1.5: )
Shifting token integer (1.5: )
Entering state 10
Stack now 0 12 37 64 39 10
Reducing stack by rule 20 (line 155):
   $1 = token integer (1.5: )
-> $$ = nterm intExpr (1.5: )
Entering state 14
Stack now 0 12 37 64 39 14
Reducing stack by rule 5 (line 125):
   $1 = nterm intExpr (1.5: )
-> $$ = nterm expr (1.5: )
Entering state 66
Stack now 0 12 37 64 39 66
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 48 ("
")
--(end of buffer or a NUL)
--EOF (start condition 0)
Next token is token end of file (2.1: )
Reducing stack by rule 26 (line 176):
   $1 = nterm expr (1.3: )
   $2 = token * (1.4: )
   $3 = nterm expr (1.5: )
-> $$ = nterm opExpr (1.3-5: )
Entering state 18
Stack now 0 12 37 18
Reducing stack by rule 9 (line 129):
   $1 = nterm opExpr (1.3-5: )
-> $$ = nterm expr (1.3-5: )
Entering state 64
Stack now 0 12 37 64
```



```
Next token is token end of file (2.1: )
Reducing stack by rule 24 (line 174):
   $1 = nterm expr (1.1: )
   $2 = token + (1.2: )
   $3 = nterm expr (1.3-5: )
-> $$ = nterm opExpr (1.1-5: )
Entering state 18
Stack now 0 18
Reducing stack by rule 9 (line 129):
   $1 = nterm opExpr (1.1-5: )
-> $$ = nterm expr (1.1-5: )
Entering state 12
Stack now 0 12
Next token is token end of file (2.1: )
Reducing stack by rule 1 (line 117):
   $1 = nterm expr (1.1-5: )
-> $$ = nterm program (1.1-5: )
Entering state 11
Stack now 0 11
Next token is token end of file (2.1: )
Shifting token end of file (2.1: )
Entering state 36
Stack now 0 11 36
Stack now 0 11 36
Cleanup: popping token end of file (2.1: )
Cleanup: popping nterm program (1.1-5: )
(1+(2*3))
```

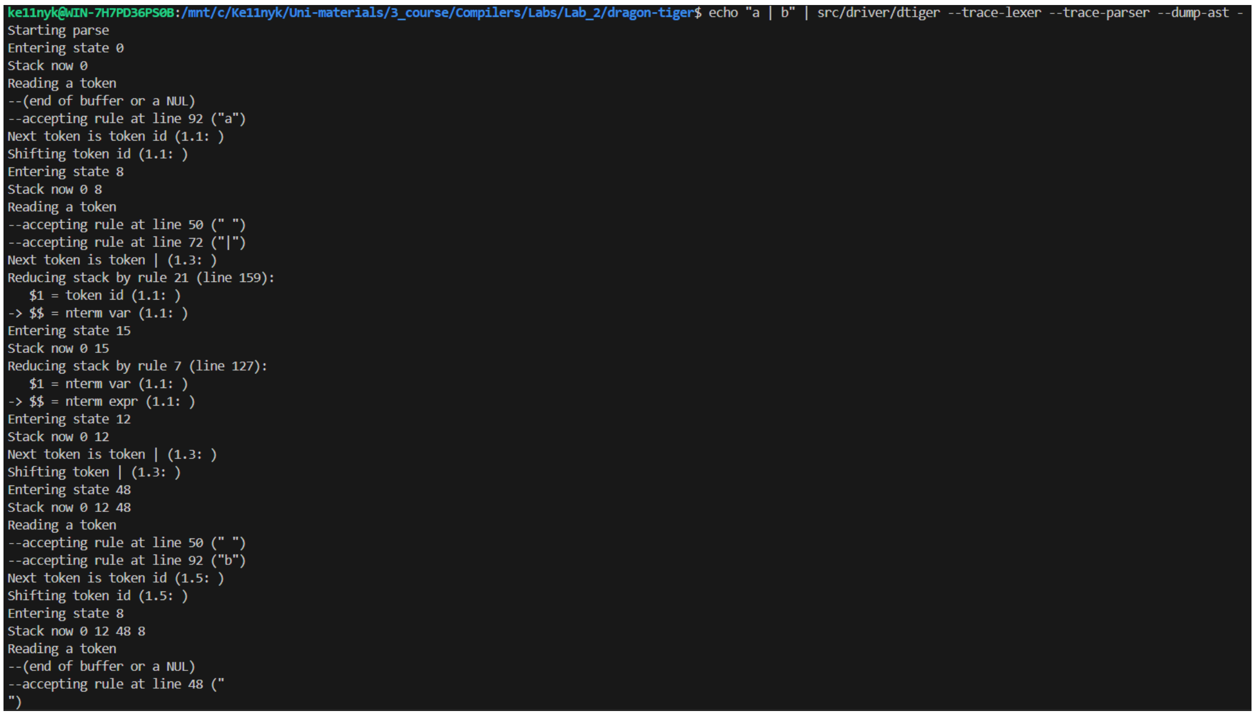


```
Entering state 14
Stack now 0 12 38 14
Reducing stack by rule 5 (line 125):
   $1 = nterm intExpr (1.5: )
-> $$ = nterm expr (1.5: )
Entering state 65
Stack now 0 12 38 65
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 48 ("
")
--(end of buffer or a NUL)
--EOF (start condition 0)
Next token is token end of file (2.1: )
Reducing stack by rule 25 (line 175):
   $1 = nterm expr (1.1-3: )
   $2 = token - (1.4: )
   $3 = nterm expr (1.5: )
-> $$ = nterm opExpr (1.1-5: )
Entering state 18
Stack now 0 18
Reducing stack by rule 9 (line 129):
   $1 = nterm opExpr (1.1-5: )
-> $$ = nterm expr (1.1-5: )
Entering state 12
Stack now 0 12
Next token is token end of file (2.1: )
Reducing stack by rule 1 (line 117):
   $1 = nterm expr (1.1-5: )
-> $$ = nterm program (1.1-5: )
Entering state 11
Stack now 0 11
Next token is token end of file (2.1: )
Shifting token end of file (2.1: )
Entering state 36
Stack now 0 11 36
Stack now 0 11 36
Cleanup: popping token end of file (2.1: )
Cleanup: popping nterm program (1.1-5: )
((5-2)-1)
```

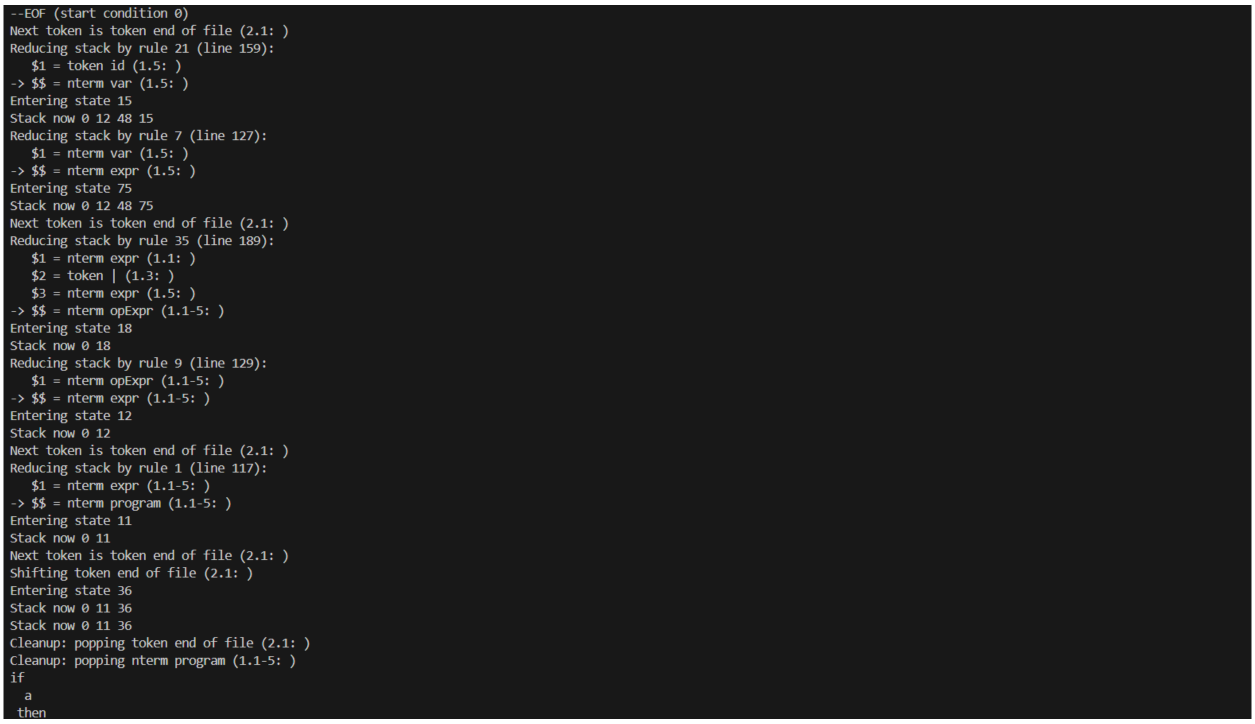


```
Entering state 14
Stack now 0 12 37 14
Reducing stack by rule 5 (line 125):
   $1 = nterm intExpr (1.4: )
-> $$ = nterm expr (1.4: )
Entering state 64
Stack now 0 12 37 64
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 48 ("
")
--(end of buffer or a NUL)
--EOF (start condition 0)
Next token is token end of file (2.1: )
Reducing stack by rule 24 (line 174):
   $1 = nterm expr (1.1-2: )
   $2 = token + (1.3: )
   $3 = nterm expr (1.4: )
-> $$ = nterm opExpr (1.1-4: )
Entering state 18
Stack now 0 18
Reducing stack by rule 9 (line 129):
   $1 = nterm opExpr (1.1-4: )
-> $$ = nterm expr (1.1-4: )
Entering state 12
Stack now 0 12
Next token is token end of file (2.1: )
Reducing stack by rule 1 (line 117):
   $1 = nterm expr (1.1-4: )
-> $$ = nterm program (1.1-4: )
Entering state 11
Stack now 0 11
Next token is token end of file (2.1: )
Shifting token end of file (2.1: )
Entering state 36
Stack now 0 11 36
Stack now 0 11 36
Cleanup: popping token end of file (2.1: )
Cleanup: popping nterm program (1.1-4: )
((0-4)+5)
```

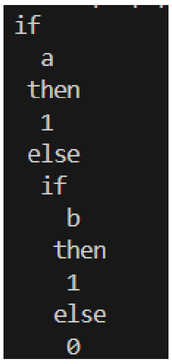
Adding support for the Boolean OR operator



```
ke11nyk@WIN-7H7PD36PS0B:/mnt/c/Ke11nyk/Uni-materials/3_course/Compilers/Labs/Lab_2/dragon-tiger$ echo "a | b" | src/driver/dtiger --trace-lexer --trace-parser --dump-ast -
Starting parse
Entering state 0
Stack now 0
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 92 ("a")
Next token is token id (1.1: )
Shifting token id (1.1: )
Entering state 8
Stack now 0 8
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 72 ("|")
Next token is token | (1.3: )
Reducing stack by rule 21 (line 159):
   $1 = token id (1.1: )
-> $$ = nterm var (1.1: )
Entering state 15
Stack now 0 15
Reducing stack by rule 7 (line 127):
   $1 = nterm var (1.1: )
-> $$ = nterm expr (1.1: )
Entering state 12
Stack now 0 12
Next token is token | (1.3: )
Shifting token | (1.3: )
Entering state 48
Stack now 0 12 48
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 92 ("b")
Next token is token id (1.5: )
Shifting token id (1.5: )
Entering state 8
Stack now 0 12 48 8
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 48 ("
")
```

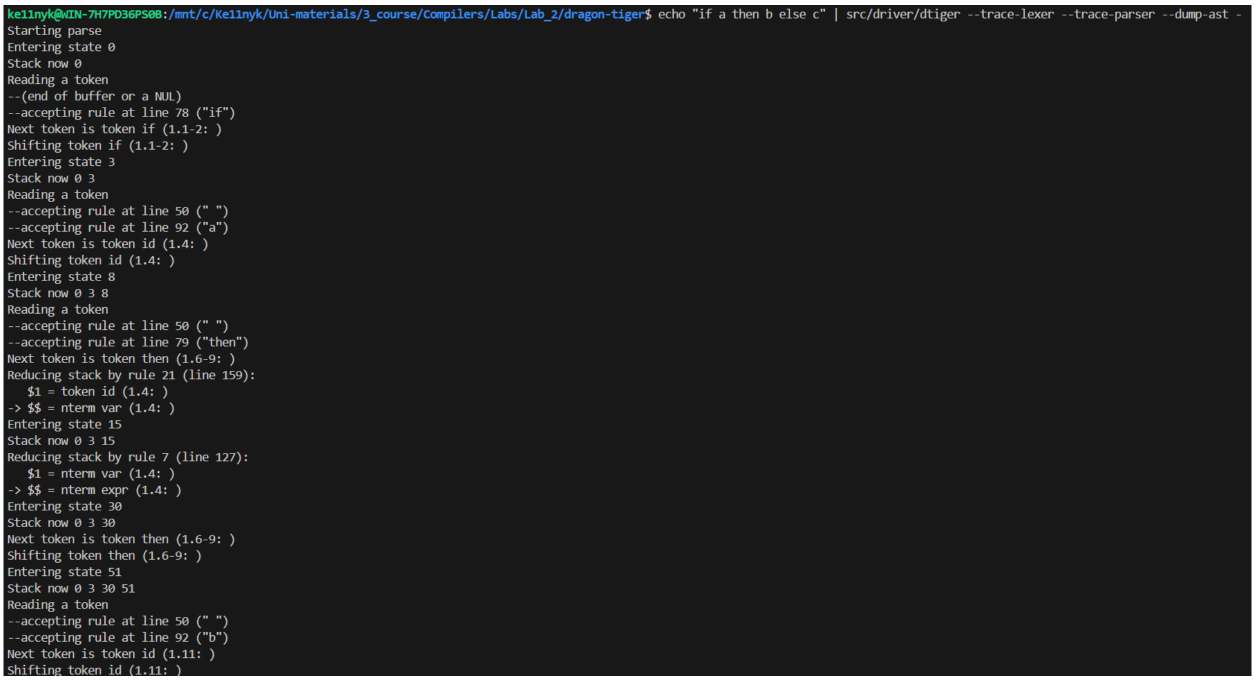


```
--EOF (start condition 0)
Next token is token end of file (2.1: )
Reducing stack by rule 21 (line 159):
   $1 = token id (1.5: )
-> $$ = nterm var (1.5: )
Entering state 15
Stack now 0 12 48 15
Reducing stack by rule 7 (line 127):
   $1 = nterm var (1.5: )
-> $$ = nterm expr (1.5: )
Entering state 75
Stack now 0 12 48 75
Next token is token end of file (2.1: )
Reducing stack by rule 35 (line 189):
   $1 = nterm expr (1.1: )
   $2 = token | (1.3: )
   $3 = nterm expr (1.5: )
-> $$ = nterm opExpr (1.1-5: )
Entering state 18
Stack now 0 18
Reducing stack by rule 9 (line 129):
   $1 = nterm opExpr (1.1-5: )
-> $$ = nterm expr (1.1-5: )
Entering state 12
Stack now 0 12
Next token is token end of file (2.1: )
Reducing stack by rule 1 (line 117):
   $1 = nterm expr (1.1-5: )
-> $$ = nterm program (1.1-5: )
Entering state 11
Stack now 0 11
Next token is token end of file (2.1: )
Shifting token end of file (2.1: )
Entering state 36
Stack now 0 11 36
Stack now 0 11 36
Cleanup: popping token end of file (2.1: )
Cleanup: popping nterm program (1.1-5: )
if
  a
 then
```

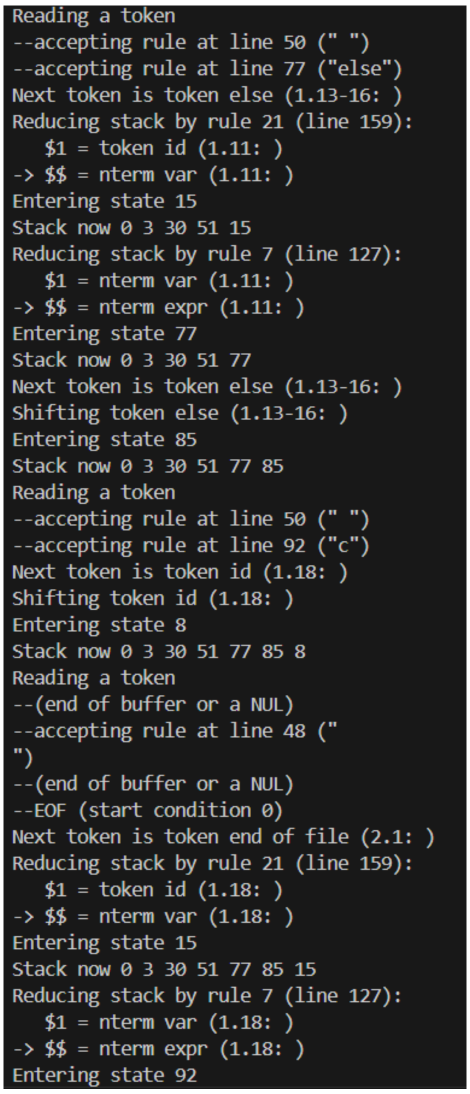


```
if
  a
 then
  1
 else
  if
    b
   then
    1
   else
    0
```

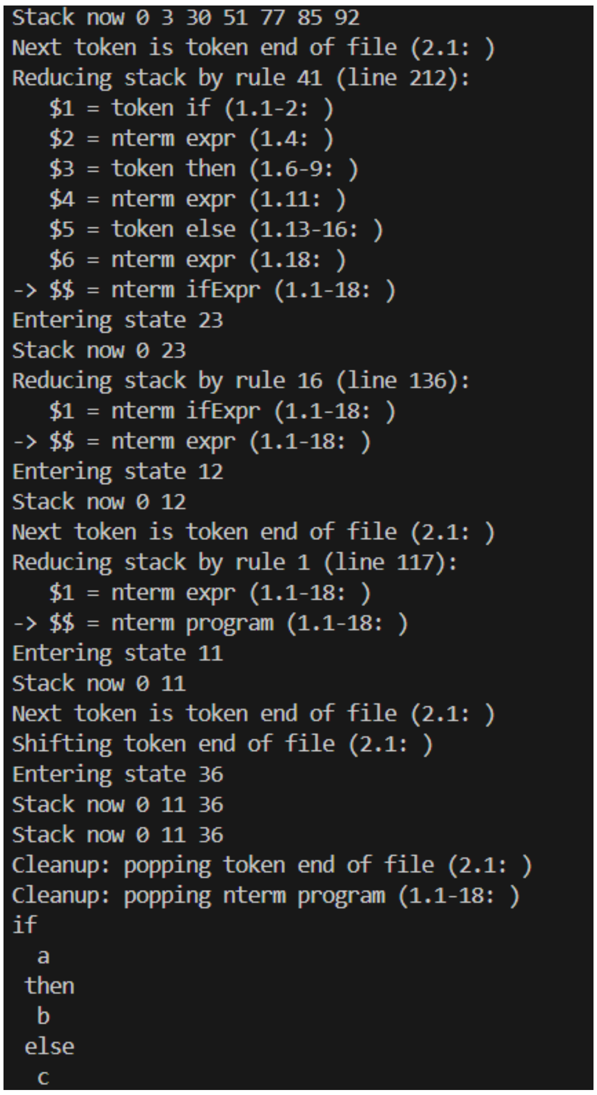
Adding support for if then else constructs



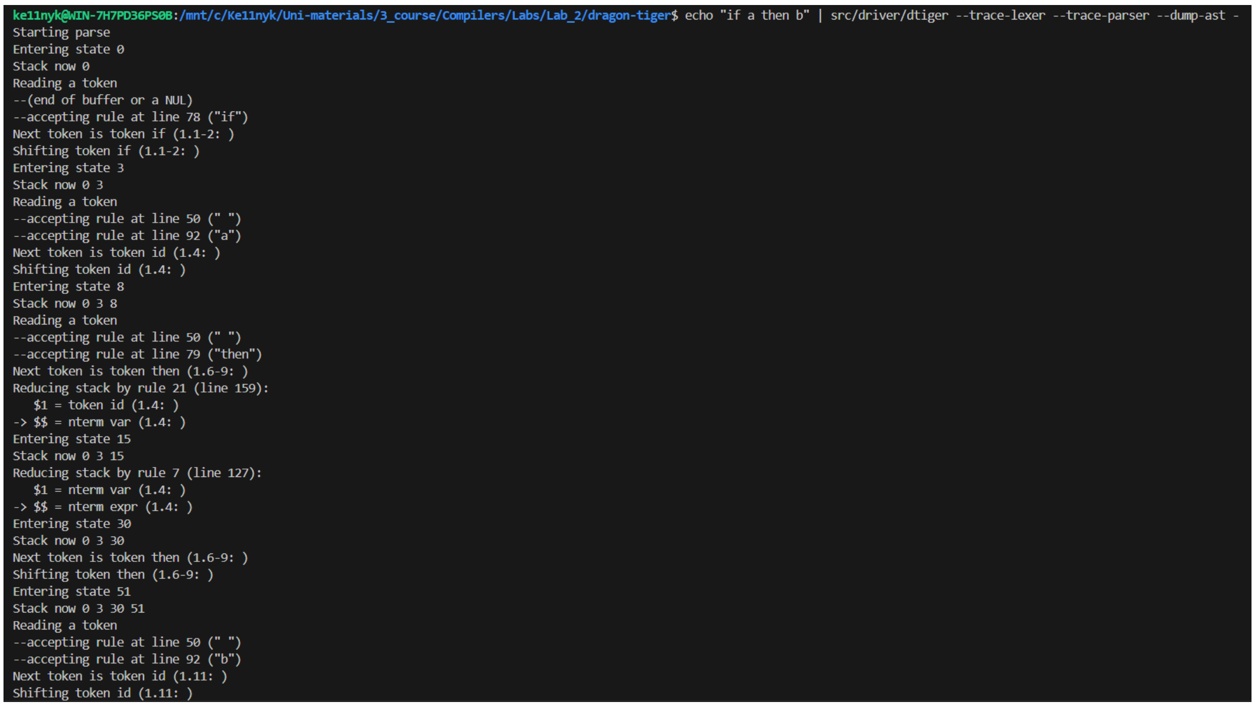
```
ke11nyk@WIN-7H7PD36PS0B:/mnt/c/Ke11nyk/Uni-materials/3_course/Compilers/Labs/Lab_2/dragon-tiger$ echo "if a then b else c" | src/driver/dtiger --trace-lexer --trace-parser --dump-ast -
Starting parse
Entering state 0
Stack now 0
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 78 ("if")
Next token is token if (1.1-2: )
Shifting token if (1.1-2: )
Entering state 3
Stack now 0 3
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 92 ("a")
Next token is token id (1.4: )
Shifting token id (1.4: )
Entering state 8
Stack now 0 3 8
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 79 ("then")
Next token is token then (1.6-9: )
Reducing stack by rule 21 (line 159):
   $1 = token id (1.4: )
-> $$ = nterm var (1.4: )
Entering state 15
Stack now 0 3 15
Reducing stack by rule 7 (line 127):
   $1 = nterm var (1.4: )
-> $$ = nterm expr (1.4: )
Entering state 30
Stack now 0 3 30
Next token is token then (1.6-9: )
Shifting token then (1.6-9: )
Entering state 51
Stack now 0 3 30 51
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 92 ("b")
Next token is token id (1.11: )
Shifting token id (1.11: )
```



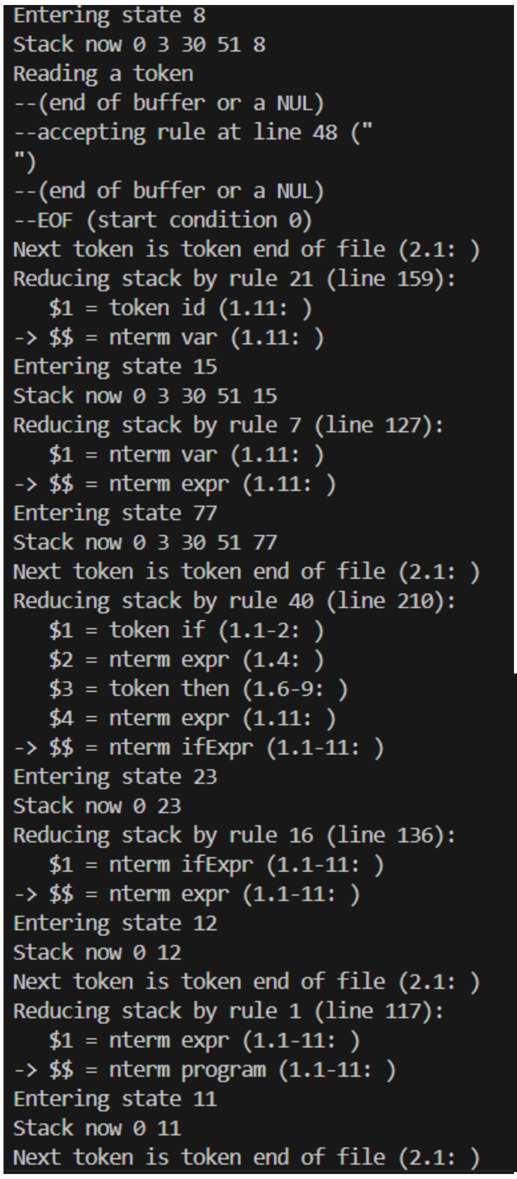
```
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 77 ("else")
Next token is token else (1.13-16: )
Reducing stack by rule 21 (line 159):
   $1 = token id (1.11: )
-> $$ = nterm var (1.11: )
Entering state 15
Stack now 0 3 30 51 15
Reducing stack by rule 7 (line 127):
   $1 = nterm var (1.11: )
-> $$ = nterm expr (1.11: )
Entering state 77
Stack now 0 3 30 51 77
Next token is token else (1.13-16: )
Shifting token else (1.13-16: )
Entering state 85
Stack now 0 3 30 51 77 85
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 92 ("c")
Next token is token id (1.18: )
Shifting token id (1.18: )
Entering state 8
Stack now 0 3 30 51 77 85 8
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 48 ("
")
--(end of buffer or a NUL)
--EOF (start condition 0)
Next token is token end of file (2.1: )
Reducing stack by rule 21 (line 159):
   $1 = token id (1.18: )
-> $$ = nterm var (1.18: )
Entering state 15
Stack now 0 3 30 51 77 85 15
Reducing stack by rule 7 (line 127):
   $1 = nterm var (1.18: )
-> $$ = nterm expr (1.18: )
Entering state 92
```



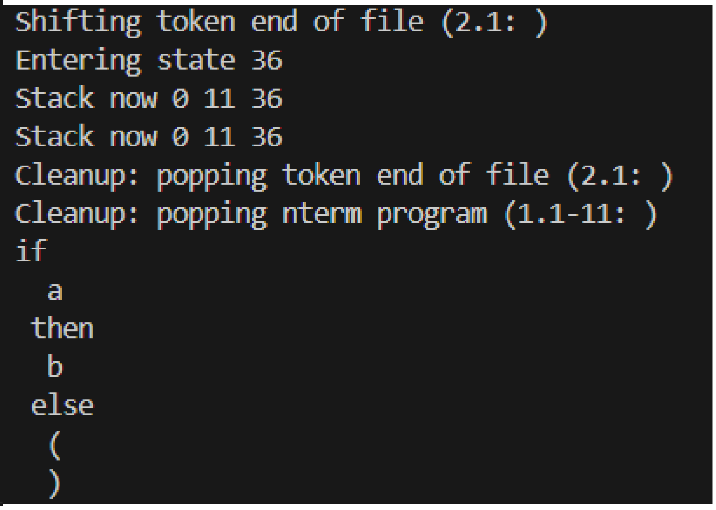
```
Stack now 0 3 30 51 77 85 92
Next token is token end of file (2.1: )
Reducing stack by rule 41 (line 212):
   $1 = token if (1.1-2: )
   $2 = nterm expr (1.4: )
   $3 = token then (1.6-9: )
   $4 = nterm expr (1.11: )
   $5 = token else (1.13-16: )
   $6 = nterm expr (1.18: )
-> $$ = nterm ifExpr (1.1-18: )
Entering state 23
Stack now 0 23
Reducing stack by rule 16 (line 136):
   $1 = nterm ifExpr (1.1-18: )
-> $$ = nterm expr (1.1-18: )
Entering state 12
Stack now 0 12
Next token is token end of file (2.1: )
Reducing stack by rule 1 (line 117):
   $1 = nterm expr (1.1-18: )
-> $$ = nterm program (1.1-18: )
Entering state 11
Stack now 0 11
Next token is token end of file (2.1: )
Shifting token end of file (2.1: )
Entering state 36
Stack now 0 11 36
Stack now 0 11 36
Cleanup: popping token end of file (2.1: )
Cleanup: popping nterm program (1.1-18: )
if
  a
 then
  b
 else
  c
```



```
ke11nyk@WIN-7H7PD36PS0B:/mnt/c/Ke11nyk/Uni-materials/3_course/Compilers/Labs/Lab_2/dragon-tiger$ echo "if a then b" | src/driver/dtiger --trace-lexer --trace-parser --dump-ast -
Starting parse
Entering state 0
Stack now 0
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 78 ("if")
Next token is token if (1.1-2: )
Shifting token if (1.1-2: )
Entering state 3
Stack now 0 3
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 92 ("a")
Next token is token id (1.4: )
Shifting token id (1.4: )
Entering state 8
Stack now 0 3 8
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 79 ("then")
Next token is token then (1.6-9: )
Reducing stack by rule 21 (line 159):
   $1 = token id (1.4: )
-> $$ = nterm var (1.4: )
Entering state 15
Stack now 0 3 15
Reducing stack by rule 7 (line 127):
   $1 = nterm var (1.4: )
-> $$ = nterm expr (1.4: )
Entering state 30
Stack now 0 3 30
Next token is token then (1.6-9: )
Shifting token then (1.6-9: )
Entering state 51
Stack now 0 3 30 51
Reading a token
--accepting rule at line 50 (" ")
--accepting rule at line 92 ("b")
Next token is token id (1.11: )
Shifting token id (1.11: )
```

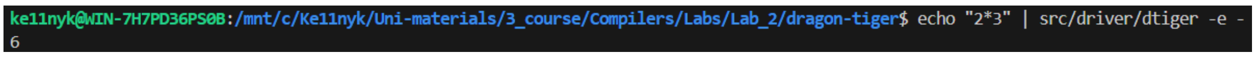


```
Entering state 8
Stack now 0 3 30 51 8
Reading a token
--(end of buffer or a NUL)
--accepting rule at line 48 ("
")
--(end of buffer or a NUL)
--EOF (start condition 0)
Next token is token end of file (2.1: )
Reducing stack by rule 21 (line 159):
   $1 = token id (1.11: )
-> $$ = nterm var (1.11: )
Entering state 15
Stack now 0 3 30 51 15
Reducing stack by rule 7 (line 127):
   $1 = nterm var (1.11: )
-> $$ = nterm expr (1.11: )
Entering state 77
Stack now 0 3 30 51 77
Next token is token end of file (2.1: )
Reducing stack by rule 40 (line 210):
   $1 = token if (1.1-2: )
   $2 = nterm expr (1.4: )
   $3 = token then (1.6-9: )
   $4 = nterm expr (1.11: )
-> $$ = nterm ifExpr (1.1-11: )
Entering state 23
Stack now 0 23
Reducing stack by rule 16 (line 136):
   $1 = nterm ifExpr (1.1-11: )
-> $$ = nterm expr (1.1-11: )
Entering state 12
Stack now 0 12
Next token is token end of file (2.1: )
Reducing stack by rule 1 (line 117):
   $1 = nterm expr (1.1-11: )
-> $$ = nterm program (1.1-11: )
Entering state 11
Stack now 0 11
Next token is token end of file (2.1: )
```

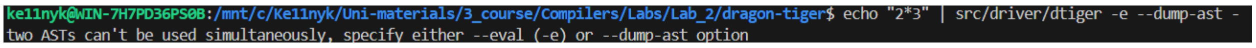


```
Shifting token end of file (2.1: )
Entering state 36
Stack now 0 11 36
Stack now 0 11 36
Cleanup: popping token end of file (2.1: )
Cleanup: popping nterm program (1.1-11: )
if
  a
 then
  b
 else
  (
  )
```

Implementing an AST evaluator for simple Tiger int expressions



```
ke11nyk@WIN-7H7PD36PS0B:/mnt/c/Ke11nyk/Uni-materials/3_course/Compilers/Labs/Lab_2/dragon-tiger$ echo "2*3" | src/driver/dtiger -e -
6
```



```
ke11nyk@WIN-7H7PD36PS0B:/mnt/c/Ke11nyk/Uni-materials/3_course/Compilers/Labs/Lab_2/dragon-tiger$ echo "2*3" | src/driver/dtiger -e --dump-ast -
two ASTs can't be used simultaneously, specify either --eval (-e) or --dump-ast option
```