Pascal-like #2

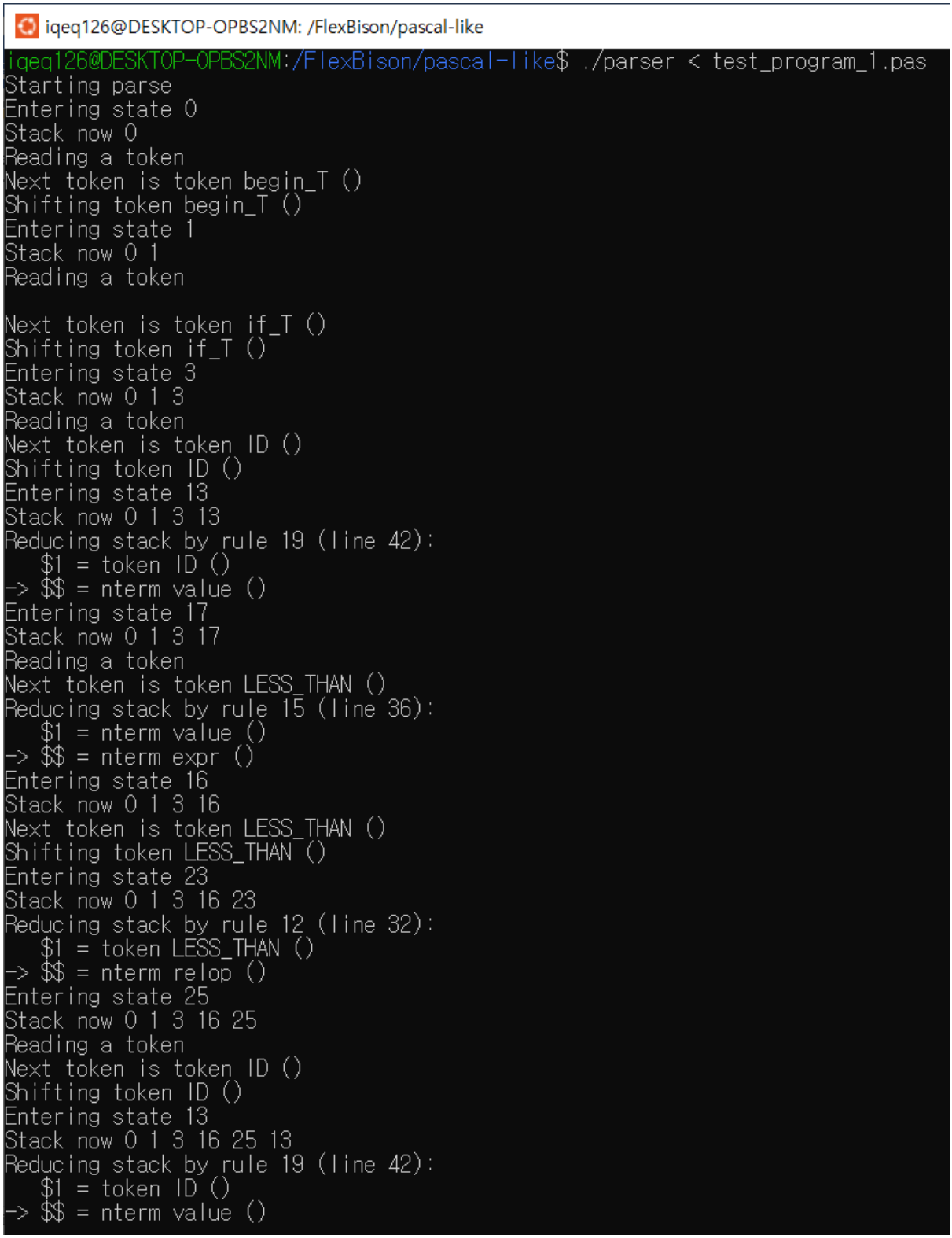
Assignment (Lab) #5

Index

1. Implement pascal-like parser

- Test the parser on test_program 1, 2 & 3

: Result of test_program 1



```
Entering state 17
Stack now 0 1 3 16 25 17
Reading a token
Next token is token then_T ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 33
Stack now 0 1 3 16 25 33
Reducing stack by rule 11 (line 30):
   $1 = nterm expr ()
   $2 = nterm relop ()
   $3 = nterm expr ()
-> $$ = nterm condition ()
Entering state 15
Stack now 0 1 3 15
Next token is token then_T ()
Shifting token then_T ()
Entering state 22
Stack now 0 1 3 15 22
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 5
Stack now 0 1 3 15 22 5
Reading a token
Next token is token ASSIGN ()
Shifting token ASSIGN ()
Entering state 19
Stack now 0 1 3 15 22 5 19
Reading a token
Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 14
Stack now 0 1 3 15 22 5 19 14
Reducing stack by rule 18 (line 41):
   $1 = token NUMBER ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 15 22 5 19 17
Reading a token

Next token is token end_T ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 30
Stack now 0 1 3 15 22 5 19 30
Reducing stack by rule 10 (line 28):
```

```
   $1 = token ID ()
   $2 = token ASSIGN ()
   $3 = nterm expr ()
-> $$ = nterm assign_statement ()
Entering state 11
Stack now 0 1 3 15 22 11
Reducing stack by rule 6 (line 21):
   $1 = nterm assign_statement ()
-> $$ = nterm statement ()
Entering state 32
Stack now 0 1 3 15 22 32
Reducing stack by rule 8 (line 24):
   $1 = token if_T ()
   $2 = nterm condition ()
   $3 = token then_T ()
   $4 = nterm statement ()
-> $$ = nterm if_statement ()
Entering state 9
Stack now 0 1 9
Reducing stack by rule 4 (line 19):
   $1 = nterm if_statement ()
-> $$ = nterm statement ()
Entering state 8
Stack now 0 1 8
Next token is token end_T ()
Reducing stack by rule 2 (line 16):
   $1 = nterm statement ()
-> $$ = nterm statements ()
Entering state 7
Stack now 0 1 7
Next token is token end_T ()
Shifting token end_T ()
Entering state 20
Stack now 0 1 7 20
Reducing stack by rule 1 (line 14):
   $1 = token begin_T ()
   $2 = nterm statements ()
   $3 = token end_T ()
-> $$ = nterm program ()
Entering state 2
Stack now 0 2
Reading a token

Now at end of input.
Shifting token "end of file" ()
Entering state 12
Stack now 0 2 12
Stack now 0 2 12
Cleanup: popping token "end of file" ()
Cleanup: popping nterm program ()
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$
```

: Result of test_program 2

```
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ ./parser < test_program_2.pas
Starting parse
Entering state 0
Stack now 0
Reading a token
Next token is token begin_T ()
Shifting token begin_T ()
Entering state 1
Stack now 0 1
Reading a token

Next token is token if_T ()
Shifting token if_T ()
Entering state 3
Stack now 0 1 3
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 3 13
Reducing stack by rule 19 (line 42):
   $1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 17
Reading a token
Next token is token LESS_THAN ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 16
Stack now 0 1 3 16
Next token is token LESS_THAN ()
Shifting token LESS_THAN ()
Entering state 23
Stack now 0 1 3 16 23
Reducing stack by rule 12 (line 32):
   $1 = token LESS_THAN ()
-> $$ = nterm relop ()
Entering state 25
Stack now 0 1 3 16 25
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 3 16 25 13
Reducing stack by rule 19 (line 42):
   $1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 16 25 17
Reading a token
Next token is token then_T ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
```

```
-> $$ = nterm expr ()
Entering state 33
Stack now 0 1 3 16 25 33
Reducing stack by rule 11 (line 30):
   $1 = nterm expr ()
   $2 = nterm relop ()
   $3 = nterm expr ()
-> $$ = nterm condition ()
Entering state 15
Stack now 0 1 3 15
Next token is token then_T ()
Shifting token then_T ()
Entering state 22
Stack now 0 1 3 15 22
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 5
Stack now 0 1 3 15 22 5
Reading a token
Next token is token ASSIGN ()
Shifting token ASSIGN ()
Entering state 19
Stack now 0 1 3 15 22 5 19
Reading a token
Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 14
Stack now 0 1 3 15 22 5 19 14
Reducing stack by rule 18 (line 41):
   $1 = token NUMBER ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 15 22 5 19 17
Reading a token
Next token is token SEMICOLON ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 30
Stack now 0 1 3 15 22 5 19 30
Reducing stack by rule 10 (line 28):
   $1 = token ID ()
   $2 = token ASSIGN ()
   $3 = nterm expr ()
-> $$ = nterm assign_statement ()
Entering state 11
Stack now 0 1 3 15 22 11
Reducing stack by rule 6 (line 21):
   $1 = nterm assign_statement ()
-> $$ = nterm statement ()
Entering state 32
Stack now 0 1 3 15 22 32
Reducing stack by rule 8 (line 24):
   $1 = token if_T ()
```

```
   $2 = nterm condition ()
   $3 = token then_T ()
   $4 = nterm statement ()
-> $$ = nterm if_statement ()
Entering state 9
Stack now 0 1 9
Reducing stack by rule 4 (line 19):
   $1 = nterm if_statement ()
-> $$ = nterm statement ()
Entering state 8
Stack now 0 1 8
Next token is token SEMICOLON ()
Shifting token SEMICOLON ()
Entering state 21
Stack now 0 1 8 21
Reading a token

Next token is token end_T ()
Error : Exiting syntax error
Error: popping token SEMICOLON ()
Stack now 0 1 8
Error: popping nterm statement ()
Stack now 0 1
Error: popping token begin_T ()
Stack now 0
Cleanup: discarding lookahead token end_T ()
Stack now 0
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$
```

: Result of test_program 3

```
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ ./parser < test_program_3.pas
Starting parse
Entering state 0
Stack now 0
Reading a token
Next token is token begin_T ()
Shifting token begin_T ()
Entering state 1
Stack now 0 1
Reading a token

Next token is token if_T ()
Shifting token if_T ()
Entering state 3
Stack now 0 1 3
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 3 13
Reducing stack by rule 19 (line 42):
   $1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 17
Reading a token
Next token is token LESS_THAN ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 16
Stack now 0 1 3 16
Next token is token LESS_THAN ()
Shifting token LESS_THAN ()
Entering state 23
Stack now 0 1 3 16 23
Reducing stack by rule 12 (line 32):
   $1 = token LESS_THAN ()
-> $$ = nterm relop ()
Entering state 25
Stack now 0 1 3 16 25
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 3 16 25 13
Reducing stack by rule 19 (line 42):
   $1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 16 25 17
Reading a token
Next token is token then_T ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
```

```
-> $$ = nterm expr ()
Entering state 33
Stack now 0 1 3 16 25 33
Reducing stack by rule 11 (line 30):
   $1 = nterm expr ()
   $2 = nterm relop ()
   $3 = nterm expr ()
-> $$ = nterm condition ()
Entering state 15
Stack now 0 1 3 15
Next token is token then_T ()
Shifting token then_T ()
Entering state 22
Stack now 0 1 3 15 22
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 5
Stack now 0 1 3 15 22 5
Reading a token
Next token is token ASSIGN ()
Shifting token ASSIGN ()
Entering state 19
Stack now 0 1 3 15 22 5 19
Reading a token
Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 14
Stack now 0 1 3 15 22 5 19 14
Reducing stack by rule 18 (line 41):
   $1 = token NUMBER ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 15 22 5 19 17
Reading a token
Next token is token SEMICOLON ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 30
Stack now 0 1 3 15 22 5 19 30
Reducing stack by rule 10 (line 28):
   $1 = token ID ()
   $2 = token ASSIGN ()
   $3 = nterm expr ()
-> $$ = nterm assign_statement ()
Entering state 11
Stack now 0 1 3 15 22 11
Reducing stack by rule 6 (line 21):
   $1 = nterm assign_statement ()
-> $$ = nterm statement ()
Entering state 32
Stack now 0 1 3 15 22 32
Reducing stack by rule 8 (line 24):
   $1 = token if_T ()
```

```
   $2 = nterm condition ()
   $3 = token then_T ()
   $4 = nterm statement ()
-> $$ = nterm if_statement ()
Entering state 9
Stack now 0 1 9
Reducing stack by rule 4 (line 19):
   $1 = nterm if_statement ()
-> $$ = nterm statement ()
Entering state 8
Stack now 0 1 8
Next token is token SEMICOLON ()
Shifting token SEMICOLON ()
Entering state 21
Stack now 0 1 8 21
Reading a token

Next token is token while_T ()
Shifting token while_T ()
Entering state 4
Stack now 0 1 8 21 4
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 8 21 4 13
Reducing stack by rule 19 (line 42):
   $1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 8 21 4 17
Reading a token
Next token is token GREATER_THAN ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 16
Stack now 0 1 8 21 4 16
Next token is token GREATER_THAN ()
Shifting token GREATER_THAN ()
Entering state 24
Stack now 0 1 8 21 4 16 24
Reducing stack by rule 13 (line 33):
   $1 = token GREATER_THAN ()
-> $$ = nterm relop ()
Entering state 25
Stack now 0 1 8 21 4 16 25
Reading a token
Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 14
Stack now 0 1 8 21 4 16 25 14
Reducing stack by rule 18 (line 41):
   $1 = token NUMBER ()
-> $$ = nterm value ()
```

```
Entering state 17
Stack now 0 1 8 21 4 16 25 17
Reading a token
Next token is token do_T ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 33
Stack now 0 1 8 21 4 16 25 33
Reducing stack by rule 11 (line 30):
   $1 = nterm expr ()
   $2 = nterm relop ()
   $3 = nterm expr ()
-> $$ = nterm condition ()
Entering state 18
Stack now 0 1 8 21 4 18
Next token is token do_T ()
Shifting token do_T ()
Entering state 29
Stack now 0 1 8 21 4 18 29
Reading a token

Next token is token begin_T ()
Shifting token begin_T ()
Entering state 1
Stack now 0 1 8 21 4 18 29 1
Reading a token

Next token is token ID ()
Shifting token ID ()
Entering state 5
Stack now 0 1 8 21 4 18 29 1 5
Reading a token
Next token is token ASSIGN ()
Shifting token ASSIGN ()
Entering state 19
Stack now 0 1 8 21 4 18 29 1 5 19
Reading a token
Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 14
Stack now 0 1 8 21 4 18 29 1 5 19 14
Reducing stack by rule 18 (line 41):
   $1 = token NUMBER ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 8 21 4 18 29 1 5 19 17
Reading a token
Next token is token PLUS ()
Shifting token PLUS ()
Entering state 26
Stack now 0 1 8 21 4 18 29 1 5 19 17 26
Reducing stack by rule 16 (line 38):
   $1 = token PLUS ()
-> $$ = nterm binaryOp ()
```

```
Entering state 28
Stack now 0 1 8 21 4 18 29 1 5 19 17 28
Reading a token
Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 14
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 14
Reducing stack by rule 18 (line 41):
   $1 = token NUMBER ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 17
Reading a token
Next token is token MINUS ()
Shifting token MINUS ()
Entering state 27
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 17 27
Reducing stack by rule 17 (line 39):
   $1 = token MINUS ()
-> $$ = nterm binaryOp ()
Entering state 28
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 17 28
Reading a token
Next token is token NUMBER ()
Shifting token NUMBER ()
Entering state 14
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 17 28 14
Reducing stack by rule 18 (line 41):
   $1 = token NUMBER ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 17 28 17
Reading a token

Next token is token end_T ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 34
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 17 28 34
Reducing stack by rule 14 (line 35):
   $1 = nterm value ()
   $2 = nterm binaryOp ()
   $3 = nterm expr ()
-> $$ = nterm expr ()
Entering state 34
Stack now 0 1 8 21 4 18 29 1 5 19 17 28 34
Reducing stack by rule 14 (line 35):
   $1 = nterm value ()
   $2 = nterm binaryOp ()
   $3 = nterm expr ()
-> $$ = nterm expr ()
Entering state 30
Stack now 0 1 8 21 4 18 29 1 5 19 30
Reducing stack by rule 10 (line 28):
```

```
   $1 = token ID ()
   $2 = token ASSIGN ()
   $3 = nterm expr ()
-> $$ = nterm assign_statement ()
Entering state 11
Stack now 0 1 8 21 4 18 29 1 11
Reducing stack by rule 6 (line 21):
   $1 = nterm assign_statement ()
-> $$ = nterm statement ()
Entering state 8
Stack now 0 1 8 21 4 18 29 1 8
Next token is token end_T ()
Reducing stack by rule 2 (line 16):
   $1 = nterm statement ()
-> $$ = nterm statements ()
Entering state 7
Stack now 0 1 8 21 4 18 29 1 7
Next token is token end_T ()
Shifting token end_T ()
Entering state 20
Stack now 0 1 8 21 4 18 29 1 7 20
Reducing stack by rule 1 (line 14):
   $1 = token begin_T ()
   $2 = nterm statements ()
   $3 = token end_T ()
-> $$ = nterm program ()
Entering state 6
Stack now 0 1 8 21 4 18 29 6
Reducing stack by rule 7 (line 22):
   $1 = nterm program ()
-> $$ = nterm statement ()
Entering state 35
Stack now 0 1 8 21 4 18 29 35
Reducing stack by rule 9 (line 26):
   $1 = token while_T ()
   $2 = nterm condition ()
   $3 = token do_T ()
   $4 = nterm statement ()
-> $$ = nterm while_statement ()
Entering state 10
Stack now 0 1 8 21 10
Reducing stack by rule 5 (line 20):
   $1 = nterm while_statement ()
-> $$ = nterm statement ()
Entering state 31
Stack now 0 1 8 21 31
Reducing stack by rule 3 (line 17):
   $1 = nterm statement ()
   $2 = token SEMICOLON ()
   $3 = nterm statement ()
-> $$ = nterm statements ()
Entering state 7
Stack now 0 1 7
Reading a token
```

```
Next token is token end_T ()
Shifting token end_T ()
Entering state 20
Stack now 0 1 7 20
Reducing stack by rule 1 (line 14):
   $1 = token begin_T ()
   $2 = nterm statements ()
   $3 = token end_T ()
-> $$ = nterm program ()
Entering state 2
Stack now 0 2
Reading a token

Now at end of input.
Shifting token "end of file" ()
Entering state 12
Stack now 0 2 12
Stack now 0 2 12
Cleanup: popping token "end of file" ()
Cleanup: popping nterm program ()
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$
```

<Submission>

- **Printout machine name and username**

  (If you miss out No score will be given)

  - **$ unmane -a**
  - **$ whoami**

- **Capture your test outputs with terminal (Window키+Shift키+s) and submit it.**

  - **$ unmane -a**

```
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ uname -a
Linux DESKTOP-OPBS2NM 4.4.0-19041-Microsoft #1237-Microsoft Sat Sep 11 14:32:00 PST 2021 x86_64 x86_64 x86_64 GNU/Linux
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ sudo uname -a
Linux DESKTOP-OPBS2NM 4.4.0-19041-Microsoft #1237-Microsoft Sat Sep 11 14:32:00 PST 2021 x86_64 x86_64 x86_64 GNU/Linux
```

  - **$ whoami**

```
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ whoami
iqeq126
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ sudo whoami
root
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$
```

  - **<Submission> testfile 1, 2, 3 (under)**

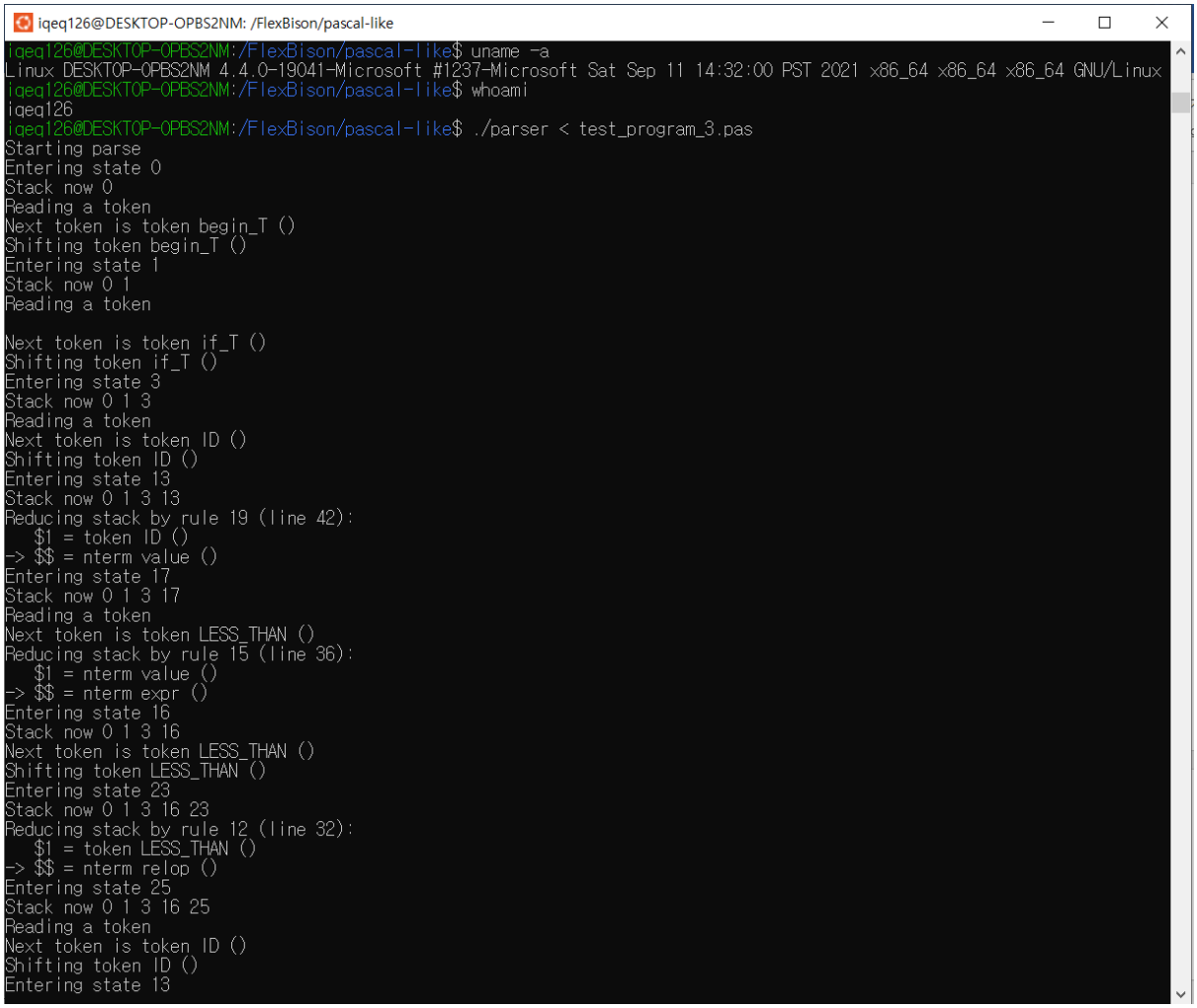
```
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ whoami
iqeq126
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ uname -a
Linux DESKTOP-OPBS2NM 4.4.0-19041-Microsoft #1237-Microsoft Sat Sep 11 14:32:00 PST 2021 x86_64 x86_64 x86_64 GNU/Linux
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ ./parser < test_program_1.pas
Starting parse
Entering state 0
Stack now 0
Reading a token
Next token is token begin_T ()
Shifting token begin_T ()
Entering state 1
Stack now 0 1
Reading a token

Next token is token if_T ()
Shifting token if_T ()
Entering state 3
Stack now 0 1 3
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 3 13
Reducing stack by rule 19 (line 42):
   $1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 17
Reading a token
Next token is token LESS_THAN ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 16
Stack now 0 1 3 16
Next token is token LESS_THAN ()
```

iqeq126@DESKTOP-OPBS2NM: /FlexBison/pascal-like

```
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ uname -a
Linux DESKTOP-OPBS2NM 4.4.0-19041-Microsoft #1237-Microsoft Sat Sep 11 14:32:00 PST 2021 x86_64 x86_64 x86_64 GNU/Linux
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ whoami
iqeq126
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ ./parser < test_program_2.pas
Starting parse
Entering state 0
Stack now 0
Reading a token
Next token is token begin_T ()
Shifting token begin_T ()
Entering state 1
Stack now 0 1
Reading a token

Next token is token if_T ()
Shifting token if_T ()
Entering state 3
Stack now 0 1 3
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 3 13
Reducing stack by rule 19 (line 42):
   $1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 17
Reading a token
Next token is token LESS_THAN ()
Reducing stack by rule 15 (line 36):
   $1 = nterm value ()
-> $$ = nterm expr ()
Entering state 16
Stack now 0 1 3 16
Next token is token LESS_THAN ()
Shifting token LESS_THAN ()
Entering state 23
Stack now 0 1 3 16 23
Reducing stack by rule 12 (line 32):
   $1 = token LESS_THAN ()
-> $$ = nterm relop ()
Entering state 25
Stack now 0 1 3 16 25
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
```


iqeq126@DESKTOP-OPBS2NM: /FlexBison/pascal-like
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ uname -a
Linux DESKTOP-OPBS2NM 4.4.0-19041-Microsoft #1237-Microsoft Sat Sep 11 14:32:00 PST 2021 x86_64 x86_64 x86_64 GNU/Linux
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ whoami
iqeq126
iqeq126@DESKTOP-OPBS2NM:/FlexBison/pascal-like$ ./parser < test_program_3.pas
Starting parse
Entering state 0
Stack now 0
Reading a token
Next token is token begin_T ()
Shifting token begin_T ()
Entering state 1
Stack now 0 1
Reading a token
Next token is token if_T ()
Shifting token if_T ()
Entering state 3
Stack now 0 1 3
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13
Stack now 0 1 3 13
Reducing stack by rule 19 (line 42):
$1 = token ID ()
-> $$ = nterm value ()
Entering state 17
Stack now 0 1 3 17
Reading a token
Next token is token LESS_THAN ()
Reducing stack by rule 15 (line 36):
$1 = nterm value ()
-> $$ = nterm expr ()
Entering state 16
Stack now 0 1 3 16
Next token is token LESS_THAN ()
Shifting token LESS_THAN ()
Entering state 23
Stack now 0 1 3 16 23
Reducing stack by rule 12 (line 32):
$1 = token LESS_THAN ()
-> $$ = nterm relop ()
Entering state 25
Stack now 0 1 3 16 25
Reading a token
Next token is token ID ()
Shifting token ID ()
Entering state 13

- Sequence Explanation

----

pascal-like.l

↓ flex pascal-like.l

lex.yy.c

↓ gcc –o pascal-like lex.yy.c –DPRINT

pascal-like

----

cp pascal-like.bnf pascal-like.l

flex pascal-like.l

⇨ lex.yy.c -#include “lex.yy.c”-> (pascal-like.y)

bison pascal-like.y (pascal-like.tab.c)

gcc –o parser pascal-like.tab.c –lfl

⇨ parser (ver.1)

cp arith.c pascal-like.c

gcc –o parser pascal-like.tab.c pascal-like.c –lfl –DYYDEGUG

⇨ parser (ver.2)

- **Explanation**
  1. **파싱을 진행합니다.**
  2. **스택에 파싱한 토큰들을 쌓아둡니다.**
  3. **토큰을 읽고 연산하는 과정을 반복합니다.**
  4. **end of file 토큰과 마주하면 프로그램을 종료합니다.**