

Mock Exam

Compiler, 2022, Term 2

[Submission Items]

1. A compressed (zip file) - The source files (Copy from terminal & paste)
2. A word file with screen dumps of - Compile and link processes in the terminal - More than 3 test outputs

[Lab] Implement a scanner and a parser of an advanced calculator working with parentheses (and) with flex & bison.

1. The source file

- scanner.l : 문자를 스캔하기 위한 scanner.l 파일

```
iqeq126@DESKTOP-OPBS2NM: /FlexBison/Calculator2/Calculator
%{
#include "parser.tab.h"
}%

%%
+ { return ADD; }
- { return SUB; }
* { return MUL; }
/ { return DIV; }
| { return ABS; }
( { return OP; }
) { return CP; }
[0-9]+ { yylval = atoi(yytext); return NUMBER; }
\n { return EOL; }
[ \t] { /* ignore whitespace */ }
{ printf("Mystery character %c\n", *yytext); }
```

- parser.y : 계산을 처리하기 위한 parser.y 파일

```
iqeq126@DESKTOP-OPBS2NM: /FlexBison/Calculator2/Calculator
/* simplest version of calculator */
%{
#include <stdio.h>
int yylex();
void yyerror(char *);
%}

/* declare tokens */
%token NUMBER
%token OP ADD SUB MUL DIV ABS OP CP
%token EOL

%%
calclist:
| calclist exp EOL { printf( " = %d\n", $2); }

exp: factor
| exp ADD factor { $$ = $1 + $3; }
| exp SUB factor { $$ = $1 - $3; }

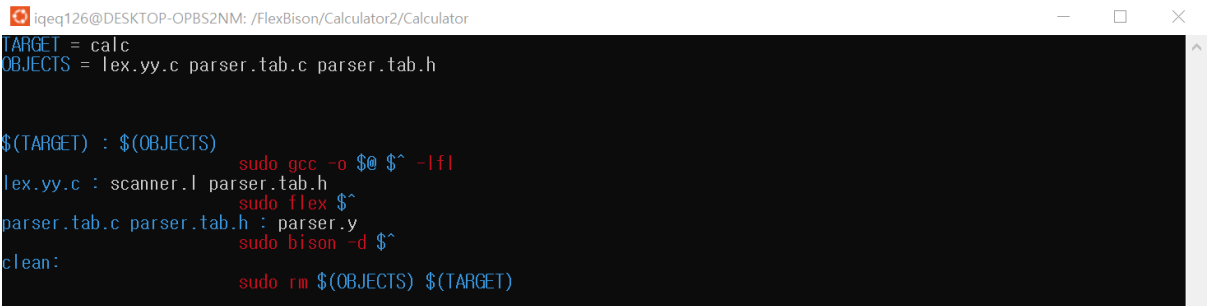
factor: term
| factor MUL term { $$ = $1 * $3; }
| factor DIV term { $$ = $1 / $3; }

term: NUMBER
| ABS term { $$ = $2 >= 0 ? $2 : - $2; }
| OP exp CP { $$ = $2; }

int main(int argc, char **argv)
{
    yyparse();
}

void yyerror(char *s)
{
    fprintf(stderr, "error: %s\n", s);
}
```

- MakeFile : 명령을 간소화하기 위한 MakeFile

A terminal window with a dark background and light blue text. The title bar shows the user 'iqeq126@DESKTOP-OPBS2NM' and the path '/FlexBison/Calculator2/Calculator'. The terminal displays the following Makefile content:

```
TARGET = calc
OBJECTS = lex.yy.c parser.tab.c parser.tab.h

$(TARGET) : $(OBJECTS)
    sudo gcc -o $@ $^ -lfl
lex.yy.c : scanner.l parser.tab.h
    sudo flex $^
parser.tab.c parser.tab.h : parser.y
    sudo bison -d $^
clean:
    sudo rm $(OBJECTS) $(TARGET)
```

명령 형식 예

: make clean => lex.yy.c, parser.tab.c, parser.tab.h, calc 삭제

: make parser.tab.c parser.tab.h => parser.y 로 parser.tab.c parser.tab.h 생성

: make lex.yy.c => scanner.l 과 parser.tab.h 로 lex.yy.c 생성

: make calc => lex.yy.c parser.tab.c parser.tab.h 로 calc 오브젝트 파일 생성

2. A word file with screen dumps of - Compile and link processes in the terminal

- Compile and link process in the terminal

```
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ ls
Makefile calc lex.yy.c parser.tab.c parser.tab.h parser.y scanner.l
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ sudo vi Makefile
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ make clean
sudo rm lex.yy.c parser.tab.c parser.tab.h calc
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ ls
Makefile parser.y scanner.l
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ make parser.tab.c parser.tab.h
sudo bison -d parser.y
make: 'parser.tab.h' is up to date.
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ ls
Makefile parser.tab.c parser.tab.h parser.y scanner.l
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ make lex.yy.c
sudo flex scanner.l parser.tab.h
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ ls
Makefile lex.yy.c parser.tab.c parser.tab.h parser.y scanner.l
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ make calc
sudo gcc -o calc lex.yy.c parser.tab.c parser.tab.h -lfl
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ ls
Makefile calc lex.yy.c parser.tab.c parser.tab.h parser.y scanner.l
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ ./calc
1+2
= 3
(1+2)*3
= 9
(1+2+(3-11))*3
= -15
3*((10-6)/2+3)
= 15
```

- Test Outputs

```
i126@DESKTOP-OPBS2NM:/FlexBison/Calculator2/Calculator$ ./calc
(1+2)*3
= 9
(1+2+(3-11))*3
= -15
3*((10-6)/2+3)
= 15
```

과정 설명.

먼저 calc는 scanner.l에서 사칙 연산자(+, -, /, *), 대입연산자(=), 괄호((,)) 문자를 각각 ADD, SUB, OP매개, CP 등의 매개변수로 두고 parser에서 그를 감지해 NUMBER -> term -> factor의 우선순위로 연산할 수 있도록 파싱한 결과를 출력하는 파일입니다.

연산 과정을 보면 (1+2)*3을 하면서는 1+2의 결과를 NUMBER로 인해 1+2를 *3보다 먼저 연산을 하게 되어 9를 출력하고,

(1+2((3-11))*3의 경우에도 마찬가지로 3-11, 1+2+, *3의 순으로 연산이 이루어져 (1+2+3-11)*3 = (-5)*3 = -15가 되어 -15를 출력하고

3*((10-6)/2+3)의 경우에도 마찬가지로 3*((4)/2+3) = 3*(2+3) = 3*5 = 15가 되어 15를 출력하게 됩니다.