SCS 2312 - Computational Models and Programming Language Concepts

Take Home Assignment
R.Jananganan
23000686

1) Context- Free Grammer (CFG)

<program>	→	<statement_list>
<statement_list>	→	<statement> <statement_list> | ε
<statement>	→	<declaration> | <print_statement>
<declaration>	→	int <identifier> = <expression> ;
<print_statement>	→	print (<identifier>);
<expression>	→	<term> <expression'>
<expression'>	→	+ <term> <expression'> | - <term> <expression'> | ε
<term>	→	<factor> <term'>
<term'>	→	* <factor> <term'> | / <factor> <term'> | ε
<factor>	→	<identifier> | <number>
<identifier>	→	letter (letter | digit)*
<number>	→	digit +

This grammar explains about the Context-Free Grammar of the built parser.c file. This c programming file supports variable declarations, arithmetic operations with proper operator precedence and print statements also.

2) Tokenizer Implementation

Here the function fgetc() is used in the code for the tokenizer to read the input file character by character, so that it classifies input into tokens such as keywords, identifiers, numbers, operators and symbols. Blank whitespaces are ignored by using the isspace() function. isalpha() and isalnum() functions are used to identify the keywords and identifiers and then isdigit() function is used to detect numbers. And finally the function ungetc() is used to push back the characters that doesn't belong to the current token into the code again so that it can be detected and read in the next step.

3) Parser

The parser is implemented in such a way that each non-terminal elements in the CFG corresponds to a function such as parseExpression(), parseTerm() and parseFactor(). The advance() function is used to retrieve the next token and expect() function is used to ensure that the correct token appears according to the grammar. If an unexpected token is met then a syntax error will be generated accordingly.

4) Semantic Analysis

A symbol tyable is implemented using arrays to store variable names and their corresponding values. The function getVar() ensures variables are declared before use. Semantic checks include detection of undeclared variables, devision by zero and optional redeclaration errors.

5) Execution & Evaluation

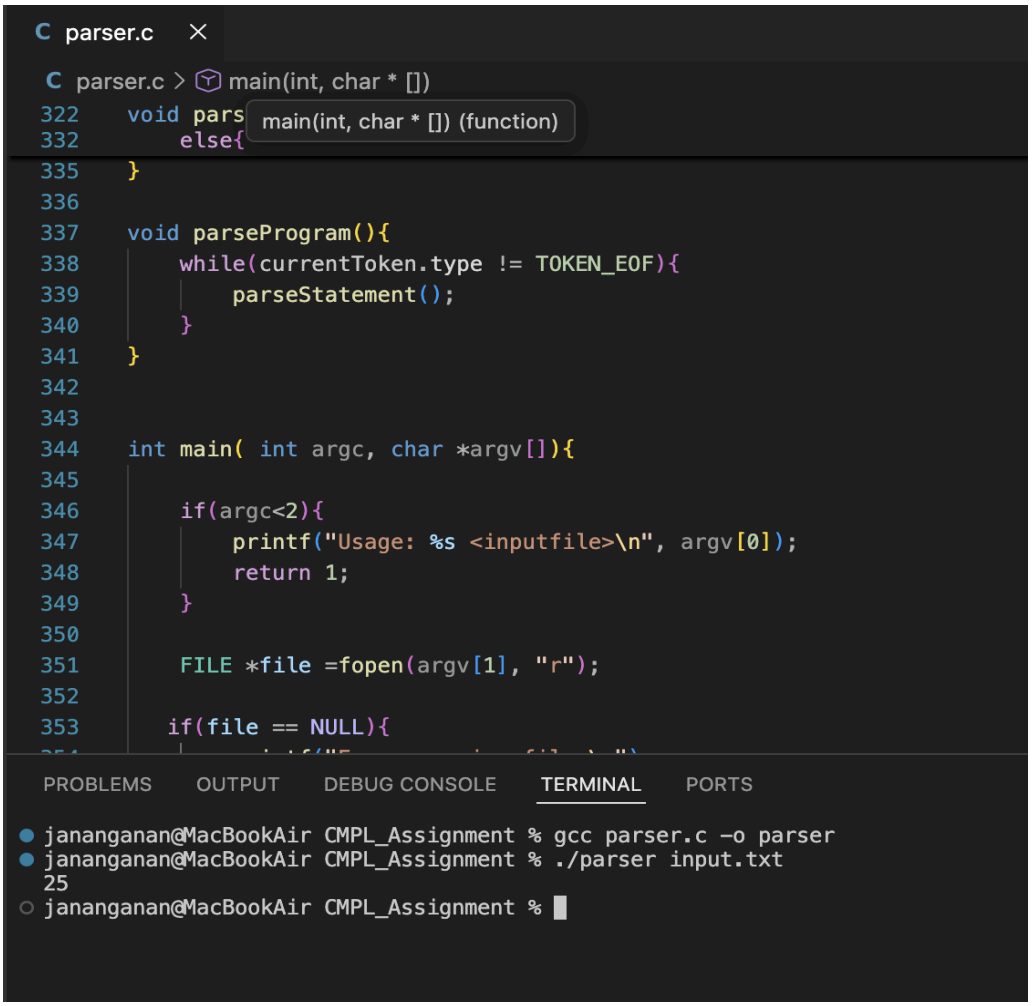
Its done during parsing. Operator precedence is maintained by separating expression parsing into three levels such as:

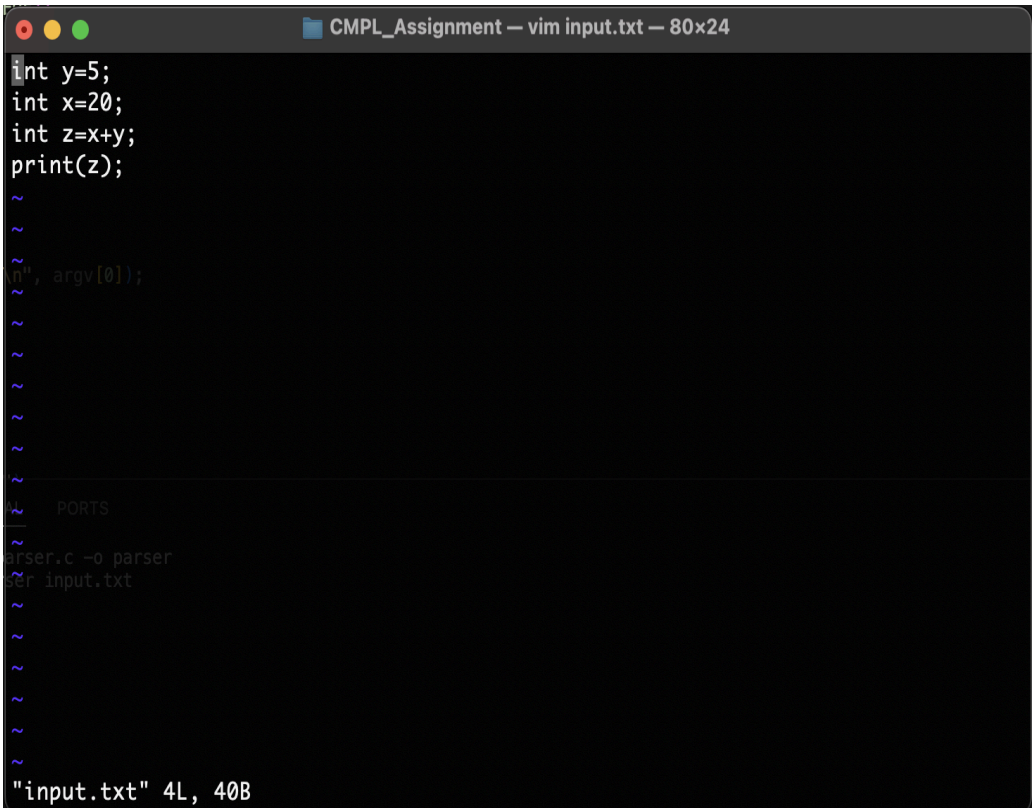
a) expression(+ , -)
b) term(*, /)
c) factor(identifier or number)

This ensures correct mathematical evaluation

6) Testing

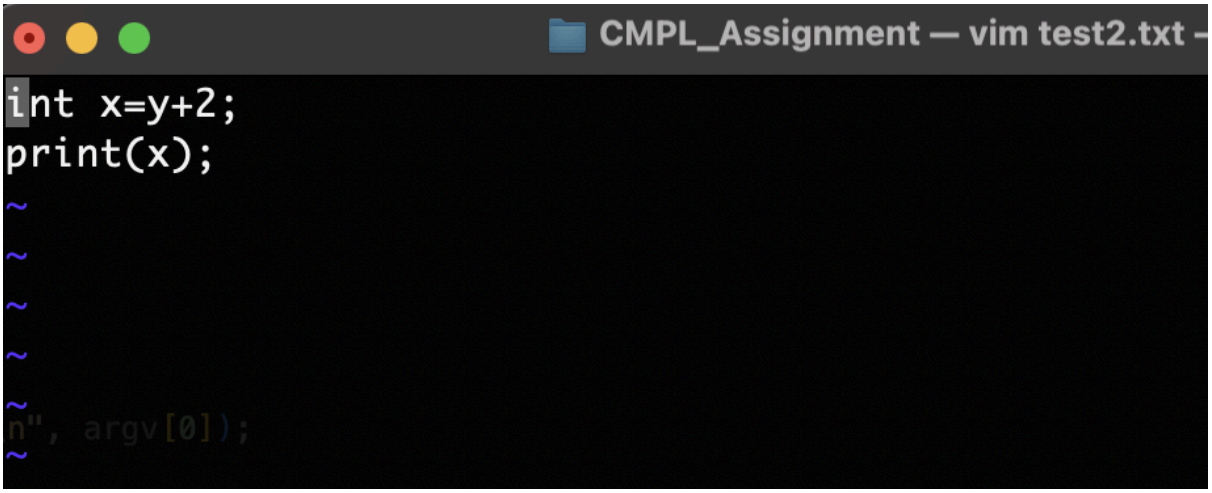
i) For correct input

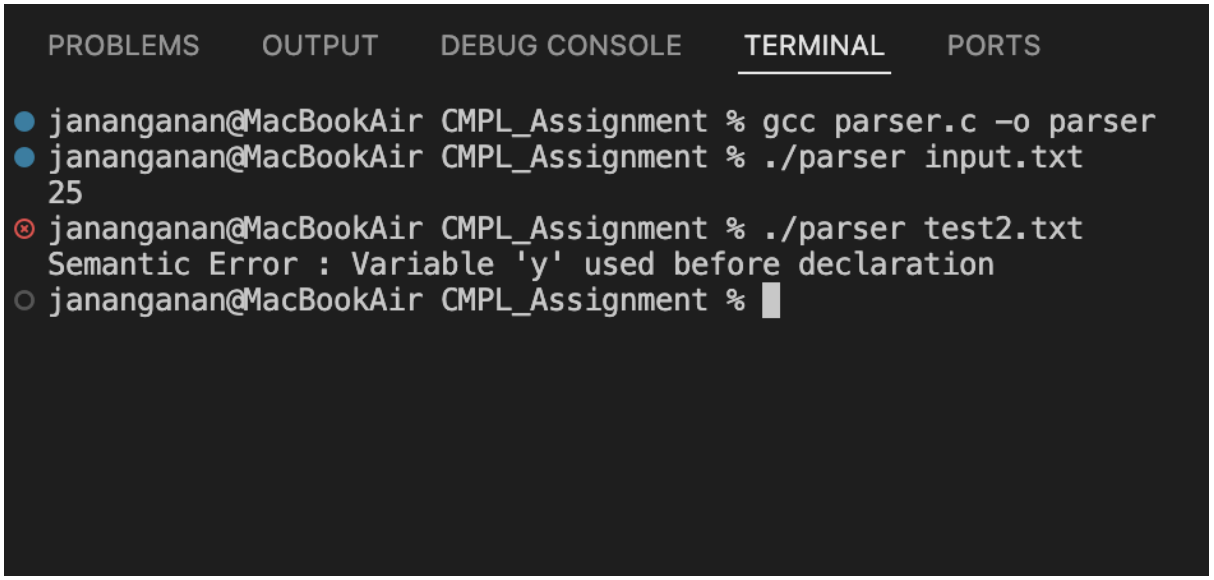

parser.c
parser.c > main(int, char * [])
main(int, char * []) (function)
void pars
else{
}
void parseProgram(){
while(currentToken.type != TOKEN_EOF){
parseStatement();
}
}
int main(int argc, char *argv[]){
if(argc<2){
printf("Usage: %s <inputfile>\n", argv[0]);
return 1;
}
FILE *file =fopen(argv[1], "r");
if(file == NULL){
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
jananganan@MacBookAir CMPL_Assignment %


CMPL_Assignment — vim input.txt — 80×24
int y=5;
int x=20;
int z=x+y;
print(z);
"input.txt" 4L, 40B

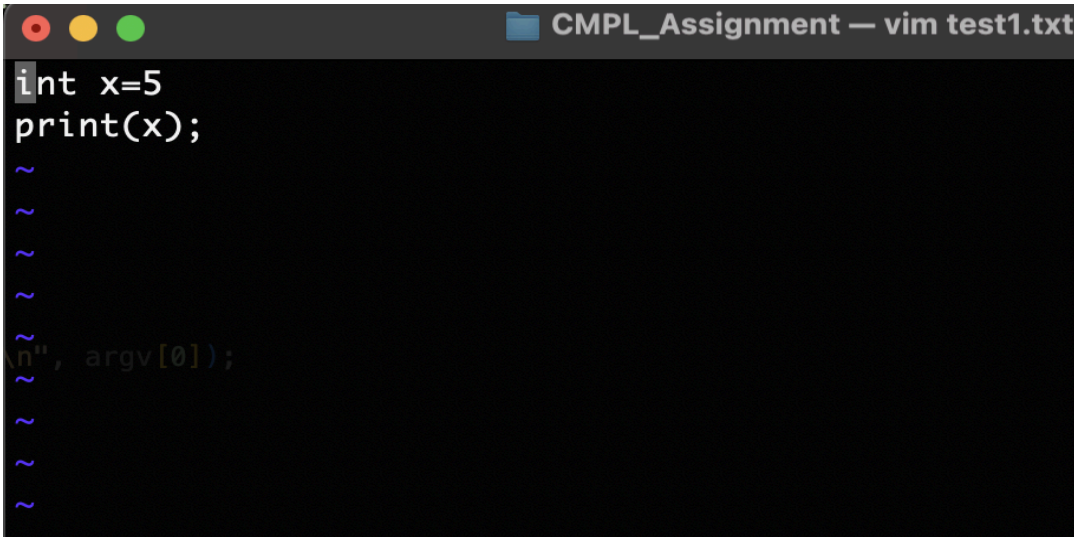
ii) For Error examples

a) **Int x = y + 2;**





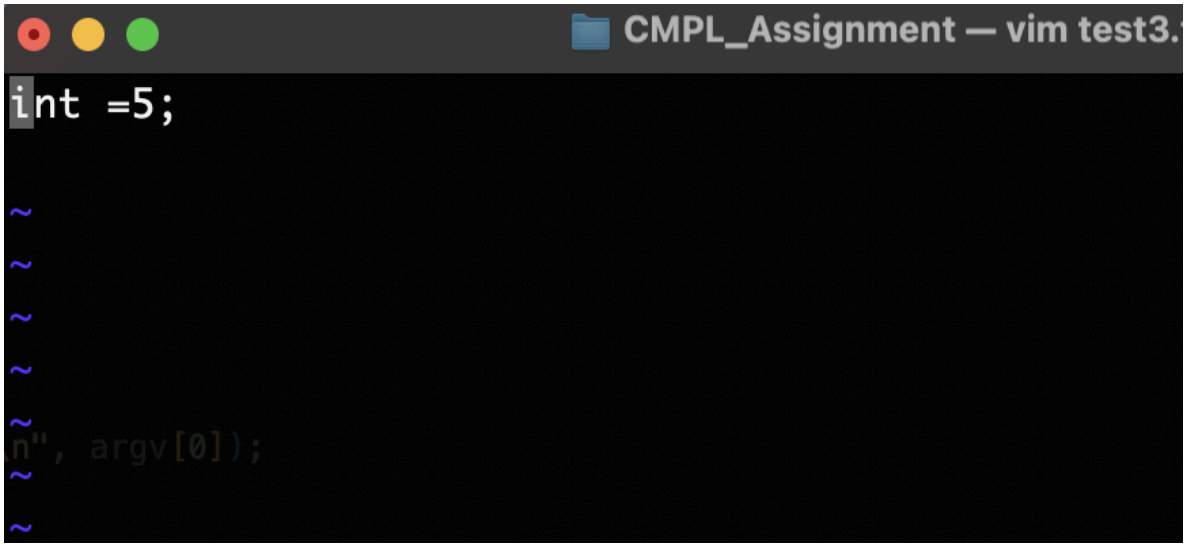
b) **int x=5**
print(x);



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
jananganan@MacBookAir CMPL_Assignment % ./parser test2.txt
Semantic Error : Variable 'y' used before declaration
jananganan@MacBookAir CMPL_Assignment % ./parser test1.txt
Syntax Error : Expected ; but found TOKEN_PRINT
jananganan@MacBookAir CMPL_Assignment %
```

c) int = 5;




```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

jananganan@MacBookAir CMPL_Assignment % gcc parser.c -o parser
jananganan@MacBookAir CMPL_Assignment % ./parser input.txt
25
jananganan@MacBookAir CMPL_Assignment % ./parser test2.txt
Semantic Error : Variable 'y' used before declaration
jananganan@MacBookAir CMPL_Assignment % ./parser test1.txt
Syntax Error : Expected ; but found TOKEN_PRINT
jananganan@MacBookAir CMPL_Assignment % ./parser test3.txt
Syntax Error: Expected identifier after int
jananganan@MacBookAir CMPL_Assignment %
```

d) **Division by zero**