CS3300: Compiler Design **Odd Sem**

Assignment #1

Scanning & Parsing Deadline: 12/08/2018, 11:55PM

Task

Write input files for **lex** and **yacc** to generate lexical analyzer and parser for the following language description.

- This language supports a subset of C statements with a few modifications.
- The language supports
  - global variables
  - function definition and function call
  - for loop, while loop
  - if and else statements
  - variable declaration and initialization
  - arrays
  - string (only of the form "..."), character, integer and floating point literals
  - single and multi-line comments
  - break, continue and return statements
- The supported data types are **int**, **float**, **double**, **void**, **char** and their pointers.
- Supported operators:
  - relational operators: $<$, $>$, $==$, $<=$, $>=$, $!=$
  - unary operators: $+$, $-$, !, $*$,&
  - binary operator: $+$, $-$, $*$, /, %
  - logical operator: &&, ||
  - assignment operators: $=$, $+=$, $-=$, $*=$, $/=$, $\%=$
- The language does **NOT** support:
  - macros, typedef, typecasting
  - function pointer
  - do-while
  - switch
  - struct, union, enum
  - bit operators: &, ~, |, ^, <<, >> and their corresponding assignment operators.
  - labels and goto
  - register, extern, static, volatile, restrict, auto
  - short, long, signed, unsigned, octal, hexadecimal, scientific notation
  - ellipsis and ternary operator

- Modifications from C:
  - The conditional expression in **while**, **if** and **for** should be the result of a relational or logical operation.
    eg.Valid:

```
if(a < b + 1 && c - 2  < d) {
    ...
}
```

    Invalid:

```
if((a && b) + c) {
    ...
}
```

  - Multiple assignment with LHS having a list of colon-separated identifiers and RHS having a list of colon-separated expressions of equal length. The operator used for this assignment is "<-".
    eg. Valid:

```
a : b : c <- c : x + 4 : a * b + c;
```

    The above is equivalent to:

```
old_a = a;
old_b = b;
old_c = c;
a = old_c;
b = x + 4;
c = old_a * old_b + old_c;
```

    Invalid:

```
a : b : c : d <- 1 : 2 : 3;
```

    No need to support pointer dereferencing or arithmetic on the LHS for multiple assignment.
  - Exponential operator ^^
    eg.

```
x = a ^^ b;  // represents x = a pow b
```

Note: The C keywords that are not a part of this language description can be used as identifiers. Any C construct not specified in the allowed constructs list is unsupported and the program should print Invalid.

Input

The input to the parser will be a program which may or may not be valid according to the above language description.
Sample execution format

```
$ ./a.out < input_program.txt
```

Output

- If the program is parsed successfully, then the following should be printed:
  Valid
  Number of variables declared
  Number of if statements

Number of else statements
Number of while loops
Number of for loops
Number of function definitions
Number of function call statements

- If the program does not parse successfully, then print:
  Invalid

Submission

Submit a tar.gz file with filename as <ROLLNO>_Lab1.tar.gz (eg. CS12B043_Lab1.tar.gz) containing the following structure:

- CS12B043_Lab1 <directory>
  - *.l
  - *.y
  - Makefile

The Makefile should run **lex**, **yacc**, compile the generated code and generate an executable **a.out** file.

Sample Test Cases

- **Test case 1**
  - **Input**

```
int factorial(int a) {
    if(a <= 0) {
        return 1;
    }
    return a * factorial(a-1);
}

int main() {
    int auto;
    scanf("%d", &auto);
    printf("factorial of %d: %d\n", factorial(auto));
    return 0;
}
```

  - **Output**

```
Valid
2
1
0
0
0
2
4
```

- **Test case 2**
  - **Input**

```
int main() {
    int a = 10;
    auto b = a;
    return 0;
}
```

– **Output**

```
Invalid
```