CS F363 Compiler Construction
Assignment-2

Due date: 29 April 2023 11:59 PM **Marks : 30**

Consider a programming language, (**Mini-C**) with the following features:

1. **Type:** int, float, char
2. **Basic statements**: expression statement, assignment statements, and compound statements, declaration statement.
3. **Control and looping statements:** if statement, for statement, while statement, and switch statement. (note that nesting of these statements is also possible.)
4. **System functions**: printf and scanf statements.
5. **Other:** Variables, arrays, and function

The syntax for the above is same as in C language. The structure of the input program is also same as a C program.

Tasks:

1. **Lexical Analysis** ***[5 marks]*** - For this phase, write a LEX program that brakes the input program into tokens (identifiers, operators, numbers, keywords, punctuators). A symbol table is created with the list of tokens obtained from the input (*no need to print the symbol table*).

   Further, report if any invalid printf and scanf statements are in the given program.

   A *valid printf* statement has one of the following structure:

   - $printf(\text{“}text\text{”})$ where $text$ is a non-empty string of symbols in the language alphabet.
   - $printf(\text{“}text\text{”}, list\_of\_variables)$ where $text$ is a non-empty string of symbols in the language alphabet and $list\_of\_variables$ is a nonempty sequence of variables separated by a comma (,).

     Further, the number of variables in $list\_of\_variables$ is must be the same as the number of $\%d$, $\%c$, and $\%f$ terms in $text$.

   A *valid scanf* statement has one of the following structure:

   - $scanf(\text{“}text\text{”})$ where $text$ is a non-empty string of symbols in the language alphabet.
   - $scanf(\text{“}text\text{”}, list\_of\_variables)$ where $text$ is a non-empty string of symbols in the language alphabet and $list\_of\_variables$ is a nonempty sequence of variables separated by a comma (,) and each variable is preceded by &.

     Further, the number of variables in $list\_of\_variables$ is must be the same as the number of $\%d$, $\%c$, and $\%f$ terms in $text$.

2. **Syntax Analysis [6 marks]**- For this phase, write a YACC program that takes tokens generated after lexical analysis and ~~construct syntactically correct sentences and the syntax errors present are printed out (missing parenthesis, semicolon)~~. checks the given input program is syntactically correct or not.

3. **Semantic Analysis [8 marks]**- Extend your YACC program written in the last phase to examine semantic errors and the abstract syntax tree is printed.

   Semantic errors like type checking, undeclared variables, multiple declarations of variables are verified.

4. **Code Generation** ***[6 marks]*** - Extend your YACC program in the last phase to generate the intermediate code in three address code format.

5. **Final stage:** ***[5 marks]*** Finally, extend your YACC program to print the output of the given input program.

Instructions

1. Do the assignment in groups of size maximum of 6.
2. Deadline is 29 April 2023 11:59 PM. No extension is possible.
3. Late submissions will be allowed up to maximum 24 hrs after the deadline with penalty of 2% per each hour.
4. ~~The input/output format and submission guidelines will be updated in a week.~~

Submission guidelines

1. Only one submission per group.
2. Place the LEX and YACC programs for each task in a separate folder and name the folder with the task name.
3. Create a *readme.txt* file that contains the information of the group members and mention how to compile the programs for each task.
4. Finally, place all the folders and readme.txt inside a new folder folder (name it with BITS ID of the one who is submitting the assignment) and zip the folder and submit it.
5. Your programs must execute on Ubuntu 22.04.

Input and Output formats

1. Input will be given in a file and the name of the file will be given at the runtime.
2. You print the output on the terminal and the format for each task is given below.
3. **Lexical Analysis**: print the list of valid tokens in the given input program in the following format

line number	lexeme	token type

   Further, print if any invalid printf and scanf statements are presented in the input program along with the exact line number in the program.

4. **Syntax Analysis**: if the input program has any syntax error print "syntax error" message, otherwise print "valid input". (No need to print the type of the syntax error.)
5. **Semantic Analysis**: Print the abstract syntax tree (if the input program has no syntax errors). For the structure and further details refer the web link given on CMS.

   If a variable is used before declaration, print an error message "undeclared variable" along with the name of the variable.

   If a variable is declared more than once, print an error message "multiple declarations of a variable" along with the name of the variable.

   Further, print if there are any type mismatches in the given input program. In the case of printf and scanf statements, the list of type specifiers (format specifiers) is mapped from left to right with the list of variables.

6. **Code Generation**: print the three-address code generated in the form of **Quadruple** (see lab sheet 10 for the exact structure.)

7. **Final stage**: print the output of the given input.

All the best.