COMPILER DESIGN LABORATORY **February 24, 2024**
IIIT Ranchi

Lab Assignment 4 :
Objective : To complete the experimentation of Lab 3 and add more features to it.

General Instructions :
1. Save all your work in this laboratory session in a separate directory, say Lab4.

Practice Problems

1. Add the feature of comments, both single line and multi-line comment of C in your lex script. To get a feel of what C compiler does, run the following program under gcc and observe the output.

2. Add string literals to your script and test its performance.

Test Program :

```
#include <stdio.h>
#include <string.h>

   /* This is a one line comment */

  /* This is a 2 line
     comment */
  /*  this is almost a recursive comment
      \*  continue with second line *\/
  */
int main()
{ // Testing for string literals and behavior of C
  char s1[] = "abcd \
1234";
  char s2[] = "1234""";
  char s3[] = """";  // null string
  char s4[] = "ab""cd""ef"; // no error generates single string
                           // abcdef "" possibly means null string
   char s5[] ="abcd"    "1234";
 /* This is a 2 line
     comment */

  char* mystr = "Here is the first line.\nHere is the second line.";
  printf("%s %ld \n", s1, strlen(s1));
  printf("%s %ld \n", s2, strlen(s2));
  printf("%s %ld \n", s3, strlen(s3));
  printf("%s %ld \n", s4, strlen(s4));
  printf("%s %ld \n", s5, strlen(s5));
  printf("%s %ld \n", mystr, strlen(mystr));
 /*  this is close to a  recursive comment
      \*  continue with second line *\/
  */
  return 0;
}
```

P1. The following subset of C is the scope for this problem.

Token Type	**Lexeme**
Keyword	int if else return
Identifier	As defined in C
Number	Natural numbers
Real	Numbers with a decimal point (with or without exponent)
Relop	< <= > >= == !=
Assignop	= += −=
Leftpar	([{
Rightpar	)] }
Delimiter	; ,

Generate a scanner from your lex specifications.
(a) Examine the input file, **inp1.txt** and the desired output file, **out1.txt**, after all the tokens have been recognized in the input file.
(b) Now write the lex specifications and run the generated scanner on this input file.
(c) Modify your specifications till the outputs are identical.

P2. Add the following features to those of P1.
(a) Augment the keyword set by {float char const static while do return void}.
(b) Octal (starts with 0 followed by digits from {0,1,2,3,4,5,6,7} and Hexadecimal numbers (starts with 0x (or 0X) followed by digits from {0,1,2,3,4,5,6,7,8,9, a,b,c,d,e,f,A,B,C,D,E,F]
(c) Add the operators at precedence levels 13, 12 and 11 from the operator table provided to you.
(i)Write a lex script for the features of P1 and P2 and generate a scanner.
(ii) Create an input file with all the features included and test your scanner with this input. Identify all the errors and modify your design till the scanner functions flawlessly.

P3. It is desired to determine the number of tokens of each type that are detected in a given input file at the end of the output file produced by the generated scanner. Test your scanner with one or two sample input files.

The operators in C++ with their attributes are given in the following table.

Precedence	**Associativity**	**Arity**	**Operator**	**Function of the operator**
16	**L**	**binary**	**[]**	**array index**
15	**R**	**unary**	**++, --**	**increment, decrement**

15	**R**	**unary**	**~**	**bitwise NOT**
15	**R**	**unary**	**!**	**logical NOT**
15	**R**	**unary**	**+, -**	**unary minus, plus**
15	**R**	**unary**	***, &**	**dereference, address of**
13	**L**	**binary**	***, /, %**	**multiplicative operators**
12	**L**	**binary**	**+, -**	**arithmetic operators**
11	**L**	**binary**	**<<, >>**	**bitwise shift**
10	**L**	**binary**	**<, <=, >, >=**	**relational operators**
9	**L**	**binary**	**==, !=**	**equality, inequality**
8	**L**	**binary**	**&**	**bitwise AND**
7	**L**	**binary**	**^**	**bitwise XOR**
6	**L**	**binary**	**|**	**bitwise OR**
5	**L**	**binary**	**&&**	**logical AND**
4	**L**	**binary**	**||**	**logical OR**
3	**L**	**ternary**	**?:**	**arithmetic if**
2	**R**	**binary**	**=, *=, /=, %=, +=, -=, <<=, >>=, &=, |=, ^=**	**assignment operators,**

End of the Experiment 3