BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
HYDERABAD CAMPUS
SECOND SEMESTER 2016 – 2017
Compiler Construction (CS F363)
Lab Sheet – 2

1. Objective of this lab is
   - To experiment with different pattern matching characters
   - To work with two rules that Lex uses
   - To use the inbuilt Lex variables
2. Background:

Metacharacter	Matches
.	any character except newline
\n	newline
*	zero or more copies of the preceding expression
+	one or more copies of the preceding expression
?	zero or one copy of the preceding expression
^	beginning of line
$	end of line
a|b	a or b
(ab)+	one or more copies of ab (grouping)
"a+b"	literal "a+b" (C escapes still work)
[]	character class

Table 1: Pattern Matching Primitives

Expression	Matches
abc	abc
abc*	ab abc abcc abccc ...
abc+	abc abcc abccc ...
a(bc)+	abc abcbc abcbcbc ...
a(bc)?	a abc
[abc]	one of: a, b, c
[a-z]	any letter, a-z
[a\-z]	one of: a, -, z
[-az]	one of: -, a, z
[A-Za-z0-9]+	one or more alphanumeric characters
[\t\n]+	whitespace
[^ab]	anything except: a, b
[a^b]	one of: a, ^, b
[a|b]	one of: a, |, b
a|b	one of: a, b

Table 2: Pattern Matching Examples

Name	Function
`int yylex(void)`	call to invoke lexer, returns token
`char *yytext`	pointer to matched string
`yyleng`	length of matched string
`yylval`	value associated with token
`int yywrap(void)`	wrapup, return 1 if done, 0 if not done
`FILE *yyout`	output file
`FILE *yyin`	input file
`INITIAL`	initial start condition
`BEGIN`	condition switch start condition
`ECHO`	write matched string

Table 3: Lex Predefined Variables

Choosing between different possible matches:
When more than one pattern can match the input, lex chooses as follows:
1. The longest match is preferred.
2. Among rules that match the same number of characters, the rule that occurs earliest in the list is preferred.

Examples

Lex program to demonstrate longest matched string.(Rule of longest match)

```
%%
begin printf("Compiler");
beginning printf("Compiler Design");
%%
```

Lex program to demonstrate priority the rule.(first rule)

```
%%
begin print("Compiler");
[a-z]+ printf("Compiler Design");
%%
```

3. **Exercises to try**
   3.1 Write a Lex Program to find octal and hexadecimal numbers
   3.2 Write a LEX program to count the number of positive & negative integers and fractions
   3.3 Write a LEX program to count number of lines, words, characters and blank spaces from a c file.
   3.4 Write a Lex program to append line numbers to give c code.
   3.5 Write a Lex program to find no.of valid identifiers.
   3.6 Write a Lex program given expression is valid or not.
   3.7 Write a Lex program to count no.of empty lines from the given file.

**********That's all folks**********