Compiler Construction Labsheet-5

Topics:

1. **yyterminate**
2. **yyrestart**
3. **input and unput**
4. **yymore and yyless**
5. **Matching a Right Context:**

Use of ***yyterminate( )*** can be used in lieu of a return statement in an action. It terminates the scanner and returns a 0 to the scanner's caller, indicating "all done". By default, yyterminate() is also called when an end-of-file is encountered.

Example:

```
%{
#include <stdio.h>
%}
%%
[a-z]+ { printf("lower case\n");
ECHO;
printf("\nBegin yyterminate\n");
yyterminate();
printf("End of yyterminate\n");
}
[a-zA-Z]+ { printf("Mixed case\n"); ECHO; }
%%
main()
{yylex();
printf("Closing Bye..\n");
}
```

Use of ***YYRESTART( )***

EX:

```
%%
[a-z]+ {printf("\nLower case token = ");
ECHO; return;
}
[a-zA-Z]+ {ECHO;}
%%
main(){
yylex();
printf("\nend of first yylex:\n");
FILE *fp;
fp=fopen("input.txt","r+");
printf(" \n Begin of 2nd lex operation on file \n");
yyrestart(fp);
```

```
yylex();
printf("\nEnd of 2nd lex operation on file\n");
}
```
create a file named input.txt and type
Hello We look at all possible Problems.

```
XX@csis-bits:~/lab5$ ./a.out
Input:Hello
Hello // wait for next input
Input: good
Lower case token = good
end of first yylex:
Begin of 2nd lex operation on file
Hello We // content from input.txt
Lower case token = look
End of 2nd lex operation on file
```

lex Routines

The following macros enable you to perform special actions.

- **input()** reads another character
- **unput()** puts a character back to be read again a moment later

One way to ignore all characters between two special characters, such as between a pair of double quotation marks, is to use *input()* like this:

```
\" while (input() != '"');
```

Use of ***unput('a'):*** returns the character ***a*** back to the input stream

yymore(): Appends the next matched string to the current value of the yytext array rather than replacing the contents of the yytext array.

yyless(): trim characters from yytext and puts them back on stdin.
yyless(n) returns all but the first n characters of the current token back to the input stream, where they will be rescanned when the scanner looks for the next match. yytext and yyleng are adjusted appropriately (e.g., yyleng will now be equal to n).

Matching a Right Context:

Lex can match a right context while matching a regexp by using the notation r1/r2 where the regexp r1 will match only if the right context matches the regexp r2 . The longest match is computed using r1 and r2 concatenated together. The lexeme reported is the match for r1.

For example, consider the following lex program.

```
%%
a+/b { printf("T_AB: %s\n", yytext); }
a+/c { printf("T_AC: %s\n", yytext); }
b { printf("T_B: %s\n", yytext); }
c { printf("T_C: %s\n", yytext); }
. ECHO;
\n return 1;
%%
int main()
{
yylex();
}
```

On input: ab
output: T_AB:a
T_B:b
On input: ac
output: T_AC:a
T_C:c
On input: abc
output: T_AB:a
T_B:b
T_C:c

Read the following lex program:

```
%%
a*b+/c+ { printf("T_AB:%s\n", yytext); }
a*b*c/a { printf("T_ABC:%s\n", yytext); }
a { printf("T_A:%s\n", yytext); }
c+ { printf("T_C:%s\n", yytext); }
. ECHO;
\n return 1;
%%
int main()
{
yylex();
}
```

Predict what it will print out for the input string abcca.

Exercise Problem

1. Write a lex program that simulates a simple desktop calculator to calculate the expressions with basic operators (+, -, *, Div) on int and floats values.

2. Lex Program that Checks a vaild arithmentc expression