Compiler Construction Labsheet-6

Theory related to the use of BEGIN directive, start states and exclusive states

BEGIN:- The directive BEGIN followed by the name of the start symbol, places the scanner in the corresponding rules. Lex activates the rules using the directive BEGIN and a start condition.

Example:

To illustrate the uses of start conditions, here is a scanner that provides two different interpretations of a string like "123.456". By default, it will treat it as three tokens: the integer "123", a dot ('.'), and the integer "456". But if the string is preceded earlier in the line by the string "expect-floats" it will treat it as a single token, the floating-point number 123.456:

```
%{
#include <math.h>
%}
%s expect
%%
expect-floats BEGIN(expect);
<expect>[0-9]+"."[0-9]+ {
printf( "found a float, = %f\n",
atof( yytext ) );
}
<expect>\n {
/* that's the end of the line, so
* we need another "expect-number"
* before we'll recognize any more
* numbers
*/
BEGIN(INITIAL);
}
[0-9]+ {
printf( "found an integer, = %d\n",
atoi( yytext ) );
}
"." printf( "found a dot\n" );
%%
```

```
Input:123.98
found an integer, = 123
found a dot
found an integer, = 98
input:125.90
found an integer, = 125
found a dot
found an integer, = 90
input: expect-floats134.50
found a float, = 134.500000
input:12.70
found an integer, = 12
found a dot
found an integer, = 70
input:expect-floats 30.70
found a float, = 30.700000
input:10.56
found an integer, = 10
found a dot
found an integer, = 56
```

Example.

```
char buf[100];
char *s;
%}
%x STRING
%%
\" { BEGIN STRING; s = buf; }
<STRING>\\n { *s++ = '\n'; }
<STRING>\\t { *s++ = '\t'; }
<STRING>\\\" { *s++ = '\"'; }
<STRING>\" { *s = 0; BEGIN 0; printf("found '%s'\n", buf); }
<STRING>\n { printf("invalid string"); exit(1); }
<STRING>. { *s++ = *yytext; }
%%
```

Exclusive start state **STRING** is defined in the definition section. When the scanner detects a quote the **BEGIN** macro shifts lex into the **STRING** state. Lex stays in the **STRING** state and recognizes only patterns that begin with **<STRING>** until another **BEGIN** is executed. Thus we have a mini-environment for scanning strings. When the trailing quote is recognized we switch back to initial state 0.

```
xxx@csis-bits:~/lab5$ ./a.out
hello
hello
tab
tab
"help"
found 'help'
"\
invalid stringgururaj@csis-bits:~/lab5$ ./a.out
" help"
found ' help'
```

Exercise Problem

1. Write a lex program to count and remove comments from a C file (write the output to a new C file). (Assume that a multi-line comment does not contain */).

Content of sample.c:

```
main()
{
 // declaration
 int a, b, c;
 /* assign
 values to the
 varaibles

 */
 a=10;
 b=10;
```

```
 /* second
multi-line

\\test
 */

a=a+1; //increment a by 1;

b=b+10; //increment b by 10;

}
```

output: content of New.c:

```
main()
{

int a, b, c;

a=10;
b=10;


a=a+1;

b=b+10;

}
```