Lex Example 1

Definition

```
D           [0-9]
%%
if          printf("IF statement\n");
[a-z]+      printf("tag, value %s\n", yytext);
{D}+        printf("decimal number %s\n",yytext);
"++"        printf("unary operator\n");
"+"         printf("binary operator\n");
```

Ambiguities resolved by precedence

Sample run: `$a.out`

```
hello
1999
hello1999
```


Ambiguous Source Rules

Lex can handle ambiguous specifications. If more than one expression can match the current input:

1) The longest match is preferred

2) Among rules which matched the same number of characters, the rule given first is preferred.

Ex 1:

```
integer   keyword action ...;
[a-z]+    identifier action ...;
```

Given the inputs:

```
integers
integer
```

Ex 2: Recognize a string in single quotes.: `'first' and 'second'`

```
'.*'      /* dangerous result of above Rule 1*/
'[^.\n']*' /* OK */
```


Lex Example 2

```
%{
/* a sample bit of code */
%}
ws          [ \t]
nonws       [^ \t\n]

%%
int cc = 0, wc = 0, lc = 0;

{nonws}+    cc += yyleng; ++wc;
{ws}+       cc += yyleng;
\n          ++lc; ++cc;
<<EOF>>     {
             printf( "%8d %8d %8d\n",lc,wc,cc);
             yyterminate();
            }
%%
main() { yylex(); }
```

Definitions

Length of token just read in

End of file

Scanning routine generated by Lex

Not really required in this example