Assignment 5 Solutions to Laboratory Questions

1) What happens if you remove the [\n] lex expression and action {} from scan.l?
A syntax error is created because the line production expects a '\n' token.

2) Why does test2.c generate a syntax error when you build it out of the box?
The digits are returned to the parsers and are not part of the (a|b)* grammar.

3) Why does test3.c NOT generate a syntax error when you build it out of the box?
The input strings are all accepted by the (a|b)* grammar.

4) After implementation, explain why one of your files (yNFA.output yDFA.output) is larger?
The y.output file is the verbose debug information and the DFA grammar and has more productions than the NFA grammar. (yDFA.output = 5554, yNFA.output = 3984)

5) How many bytes are in your executable programs (hw05testNFA hw05testDFA)?
hw05testNFA = 39416, hw05testDFA = 39416, the same size executables!

6) What is a reason to write your grammar in DFA over NFA inside a yacc parser?
To remove reduce/reduce errors or change the productions to use left recursion.

7) In yacc, how do you define the starting state?
%start start_production

8) In yacc, what does an accepting state have that is not found in a transition state?
An empty production.

9) Write your yacc NFA productions for solving language [(a|b)*(a|b)b(a|b)].

```
A0              : 'a' A0
                | 'b' A0
                | 'a' A1
                | 'b' A1
                ;
A1              : 'b' A2
                ;
A2              : 'a' A3
                | 'b' A3
                ;
A3              : /* empty */
                ;
```

10) Write your yacc DFA productions for solving language [(a|b)*(a|b)b(a|b)].

```
A0              : 'a' A1
                | 'b' A4
                ;
A1              : 'a' A1
                | 'b' A2
                ;
A2              : 'a' A3
                | 'b' A5
                ;
A3              : 'a' A1
                | 'b' A2
                | /* empty */
                ;
A4              : 'a' A1
                | 'b' A2
                ;
A5              : 'a' A3
                | 'b' A5
                | /* empty */
                ;
```