Compsci 330 Week 2 Tutorial

What is the meaning, if any, of the following regular expressions in JFlex?

- happy
- "happy"
- 'happy'
- [happy]
- a-z
- [a-z]
- happy|sad
- "happy|sad"
- [happy|sad]
- ["happy"|"sad"]

Try running the program

```
package grammar;

import java.io.*;

%%

%public
%class Test
%type Void

%{
     void prompt() {
          System.out.println( "1) [happy]" );
          System.out.println( "2) a-z" );
          System.out.println( "3) [happy|sad]" );
          System.out.println( "4) \"happy|sad\"" );
          System.out.println( "5) [\"happy\"|\"sad\"]" );
          System.out.print( "Case?: " );
          }
%}
%init{
    yybegin( NORMAL ); prompt();
%init}

newline   =    \r | \n | \r\n


%state NORMAL, CASE1, CASE2, CASE3, CASE4, CASE5

%%
<NORMAL> {
     1{newline}             { yybegin( CASE1 ); System.out.print( "[happy]: " );
}
     2{newline}             { yybegin( CASE2 ); System.out.print( "a-z: " ); }
     3{newline}             { yybegin( CASE3 );
                                System.out.print( "[happy|sad]: " ); }
     4{newline}             { yybegin( CASE4 );
                                System.out.print( "\"happy|sad\": " ); }
     5{newline}             { yybegin( CASE4 );
                                System.out.print( "[\"happy\"|\"sad\"]: " ); }
     }

<CASE1>[happy]          { System.out.println( "Match " + yytext() ); }
<CASE2>a-z                  { System.out.println( "Match " + yytext() ); }
<CASE3>[happy|sad]      { System.out.println( "Match " + yytext() ); }
<CASE4>"happy|sad"      { System.out.println( "Match " + yytext() ); }
<CASE4>["happy"|"sad"]  { System.out.println( "Match " + yytext() ); }
{newline}               {
                            yybegin( NORMAL );
                            prompt();
                            }
.                           { }
```

with Main.java

```
import grammar.*;
import java.io.*;

public class Main {
     public static void main( String[] arg ) {
          try {
               Test testLex = new Test( System.in );
               testLex.yylex();
               }
          catch ( IOException exception ) {
               System.out.println( exception );
               }
          }
     }
```

Create regular expressions to match the following:

- Identifiers, that include underscores as letters.
- Identifiers without any digits.
- Symbolic IP addresses, such as www.cs.auckland.ac.nz.
- Numeric IP addresses, such as 130.216.35.35.

Write a Jflex program, and a Main class to do the following:

- Each invocation of yylex() consumes input that does not correspond to hexadecimal numbers, and does not return anything.
- When 0x or 0X is matched, changes into a state to process hexadecimal digits, matching them one character at a time.
- When processing hexadecimal digits, computes the value of the number in internal form, as it goes.
- When detecting a character that is not a hexadecimal digit, returns the value of the number processed as an int, and changes back to the normal state.
- When end of file is reached in the normal state, throws an exception.
- The main() method in the Main class creates a lexical analyser, then loops invoking yylex(), and printing the return value, until it reaches end of file.

Go over JFlex questions in previous tests.