King Saud University
College of Computer and Information Sciences
Computer Science Department

CSC 340: Programming Language and Compilation
Programming Assignment 2: Using http://hackingoff.com/compilers
Due Date: April 12, 2017

For this assignment, you will use Lex for the lexical part of the task and the URL: http://hackingoff.com/compilers for writing a parser for a language called Mython. Mython is a stripped down version of Python. Most of the interesting features have been eliminated (data structure and polymorphism); however it still contains some of the syntactic elements while having a flavor similar to Pascal.

Part I: Lexical Analysis of Mython

You will use Lex to capture the tokens of the Mython language. To do this, you need to define a constant value associated with each token, so that you can return this value every time your parser needs a new token.

- **Keywords**: def fed if fi else print while elihw input true false return
- **Punctuation elements**: ( ) : ,
- **Operators**: = (assignment) + - * / == (equality) < <= >= <>
- **Identifiers**: String of alphanumeric (and _) starting with an alphabetic character
- **Literals**: integer and strings
- **Comments**: start with ! character until the end of the line
- Mython is case sensitive.

```
Program → function_list end_list
function_list → function_list function
function_list → function
function → def ID ( parameters ) : statements fed
function → def ID ( ) : statements fed
parameters → parameters , ID
parameters → ID
statements → statements statement
statements → statement
statement → assignment_stmt
statement → print_stmt
statement → input_stmt
statement → condition_stmt
```

statement → while_stmt
statement → call_stmt
statement → return_stmt
assignment_stmt → ID = expression
return_stmt → return exp
expression → exp == exp
expression → exp <> exp
expression → exp < exp
expression → exp <= exp
expression → exp > exp
expression → exp >= exp
expression → exp
expression → (expression)
exp → exp + term
exp → exp - term
exp → term
term → term * factor
term → term / factor
term → factor
factor → (exp)
factor → INT_LITERAL
factor → STRING_LITERAL
factor → ID
factor → true
factor → false
factor → call_stmt
print_stmt → print (expression_list)
input_stmt → ID = input ( )
call_stmt → ID ( )
call_stmt → ID (expr_list)
condition_stmt → if_head statements fi
condition_stmt → if_head statements else : statements fi
if_head → if expression :
while_stmt → while expression : statements elihw
expression_list → expression_list , expression
expression_list → expression
expr_list → expr_list , exp
expr_list → exp
end_list → end_list end
end_list → end
end → call_stmt
end → print_stmt
end → input_stmt

For this assignment you need to submit the following:

1) **Token classes**
2) **A list of regular expressions describing the relevant token classes**
3) **A copy of NFA and DFA using the URL:** http://hackingoff.com/compilers
4) **Extract the needed table and translate into Java code**
5) **Several samples of the input and output**
    - **e.g., Input:**
      X1=5
    - **Output:**
      <identifier, "X1">,<int,"5">

If you want to test your code, consider the following program. Notice that this program does not cover all possible things that could be testing based on the given specification. Test your code thoroughly.

```
def f_64t(n2,y) :
name = "ali"
if ( 2+ 7+ m8j == 8*7) :
if nh <> ij :
return n
fi
else :
print ("jj",false )
fi
fed
def jk( ) :
! comment 432
while (n1s<>true):
cd= input ()
elihw
fed
jk()
```

This program should compile and have no errors