# King Saud University
# College of Computer and Information Sciences
# Computer Science Department
# CSC 340: Programming Language and Compilation
# Programming Assignment 3: Writing a Parser for Mython using an online Compiler generation tool (http://hackingoff.com/compilers)

In this assignment, you will use the online compiler generation tool (http://hackingoff.com/compilers) to write an SLR parser for Mython. The tool will help you generate the NAF and DFA for the viable prefixes of the handle and their table representation. Then you will write a Java program that uses the generate table to do the actual parser.

You will need to submit a copy of each of the following

1) NFA and DFA as generated by the compiler generation tool
2) The table representing the DFA
3) Your Java code
4) Several samples of input and output. The input is a stream of tokens (as would be generated by a lexical analyzer) representing a simple Mython program and the output is a message indicating if it is syntactically correct. Your samples should include correct and some incorrect Mython programs.

## Due Date: Wednesday 3 May 2017

**You will find below a CFG describing Mython**

```
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