

# ICS 142: Compilers and Interpreters

*Prof. Dr. Michael Franz & Dr. Andreas Gal, Fall Quarter 2007*

## Assignment 5

due at the beginning of class on December 6th

### A Compiler for the Complete Programming Language SimPL

As the final assignment, you will extend one of your previous compilers to compile the complete SimPL language. This is now a “real” programming language with all the basic features of commonly-used industrial-strength languages.

New in this language are user-defined functions, procedures and arrays. In array declarations, the length of the array is given, and elements are indexed from 0 to length-1. Function procedures are interesting because you can use them to write recursive algorithms. The difference between a “function” and a “procedure” is that the former return a result and the latter does not. Obviously, only functions are allowed to contain **return** statements, even though the grammar allows them also in proper procedures – your compiler should emit an error if a procedure tries to return a result.

For this assignment, you have a choice whether you want to generate code for the “DLX” processor as your target architecture or whether you want to target Microsoft’s Common Language Runtime (“Dot Net”).

Note: To obtain credit, **your solution must be submitted electronically by the due date**, using the instructions on the TA’s web page.

## EBNF for SimPL

letter = “a” | “b” | ... | “z”.

digit = “0” | “1” | ... | “9”.

relOp = “==” | “!=” | “<” | “<=” | “>” | “>=”.

ident = letter {letter | digit}.

number = digit {digit}.

factor = ident { “[ expression ] ” | number | “( expression )” | funcCall .

term = factor { (“\*” | “/”) factor }.

expression = term { (“+” | “-”) term }.

relation = expression relOp expression .

assignment = “let” ident “<-” expression.

funcCall = “call” ident [ “( [ expression { “,” expression } ] ) ” ].

ifStatement = “if” relation “then” statSequence [ “else” statSequence ] “fi”.

whileStatement = “while” relation “do” StatSequence “od”.

returnStatement = “return” [ expression ] .

statement = assignment | funcCall | ifStatement | whileStatement | returnStatement.

statSequence = statement { “;” statement }.

typeDecl = “var” | “array” “[ number ] ” { “[ number ] ” }.

varDecl = typeDecl ident { “,” ident } “;” .

funcDecl = (“function” | “procedure”) ident [formalParam] “;” funcBody “;” .

formalParam = “( [ ident { “,” ident } ] )”.

funcBody = { varDecl } “{” [ statSequence ] “}”.

computation = “main” [ varDecl ] { funcDecl } “{” statSequence “}” “.” .

## Predefined Function

InputNum()      read a number from the standard input

## Predefined Procedure

OutputNum(x)    write a number to the standard output

OutputNewLine() write a carriage return to the standard output