# Programming Language Technology

Exam, 17 August 2016 at 14:00 – 18:00 in M

Course codes: Chalmers DAT151, GU DIT231.
Teacher: Fredrik Lindblad, will visit around 15:00 and 16:30. Phone: 031-7722038

**Grading scale**: Max = 60p, VG = 5 = 48p, 4 = 36p, G = 3 = 24p.
**Allowed aid**: an English dictionary.

Please answer the questions in English. Questions requiring answers in code can be answered in any of: C, C++, Haskell, Java, or precise pseudocode.

For any of the six questions, an answer of roughly one page should be enough.

**Question 1 (Grammars):** Write a labelled BNF grammar that covers the following constructs in a C-like imperative language: A program is a list of statements. Statement constructs are:

- `while` statements
- block statements (lists of statements surrounded by curly braces)
- expression statements (`E;`)

Expression constructs are:

- identifiers/variables
- integer literals
- function applications (`f(E,F,..)`)
- greater-than (`E > F`)
- multiplication (`E * F`)
- pre-decrement for variables (`--x`)

Operator precedences and associativity should follow the C standard. You can use the standard BNFC categories `Integer` and `Ident` as well as list shorthands, and `terminator`, `separator` and `coercions` rules. Note that function definitions should not be part of the grammar. (10p)

**Question 2 (Trees):** Show the parse tree and the abstract syntax tree of the statement

```
while (2 * --y > x) {f(--x);}
```

in the grammar that you wrote in question 1. In the parse tree show the coercions explicitly. (10p)

**Question 3 (Typing and evaluation):**

A. Write standard typing rules or syntax-directed type-checking code (or pseudocode) for the following 5 constructs of the grammar in question 1: while-statements and expression forms variable/identifier, function application, pre-decrement and multiplication. The variable context must be made explicit. (5p)

B. Write big-step operational semantic rules or syntax-directed interpretation code (or pseudocode) for the same 5 constructs as in part A. The environment must be made explicit. (5p)

**Question 4 (Parsing):**

A. Show a BNF grammar for expressions with the constructs boolean and, subtraction, less-than, variables and parentheses. Associativity and precedence should follow the C standard. The built-in BNFC `Ident` token type may be used, but no short-hands such as `coercions`. (4p)

B. Trace the LR-parsing of the expression `x && y - z < w`. Show how the stack and the input evolves and which actions are performed. (6p)

**Question 5 (Compilation):**

A. Write compilation schemes for each of the constructs of the grammar in question 1 except function application (in total 8 statement and expression constructs). It is not necessary to remember exactly the names of the JVM instructions – only what arguments they take and how they work. (6p)

B. Give the small-step semantics of the JVM instructions you used in the compilation schemes in part A. (4p)

**Question 6 (Functional languages):** Show the big-step operational semantics rules (not as code) for a functional language with the expression constructs function application, $\lambda$-abstraction, variables, integer literals and integer multiplication. The evaluation strategy should be call-by-value. Use closures and explicit environment. (6p)

Show the derivation tree (using your operational semantics) of the evaluation of the expression

```
(\f -> f (f 5)) (\x -> 2 * x)
```

(4p)