Programming Languages

Exam, 10 March 2008

8.30–12.30 in M

Course codes: Chalmers TIN321, GU INN130

Teacher: Aarne Ranta (tel. 1082)

Grading scale: Max = 60p, VG = 5 = 48p, 4 = 36p, G = 3 = 24p.

Aids: an English dictionary.

Instructions

This exam has three groups of questions, one **easy**, one **intermediate**, and one **advanced**. Points are distributed in such a way that doing the easy questions is enough to pass the exam (mark 3 or G). All the easy and around half of the intermediate are needed to get mark 4. All the intermediate, or half of them and the advanced, are needed to get the highest mark (5 or VG).

From another perspective, the easy questions can be answered by anyone who has managed to do the labs without any further reading. The intermediate questions may require any material from the lecture notes. The advanced questions may also require material from the book, chapters 1-6, that is not covered by the lectures.

You can get points for the more difficult parts without doing the easier ones.

The bonus points from exercises are added to the total score by the teachers.

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

Text in the answers can be in any of: Bulgarian, Danish, Dutch, English, Estonian, Finnish, French, German, Italian, Norwegian, Russian, Spanish, and Swedish.

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

Group 1: easy questions

1. Write a BNF grammar that covers the following constructs of C/C++:

- `while` loops
- `if` statements with `else`
- statements formed from expressions by adding a semicolon `;`
- expressions that are identifiers or integer literals or preincrements (`++i`) or postincrements (`i++`); in the increments, the operands are identifiers

An example statement is shown in question 2. You can use the standard BNFC categories `Integer` and `Ident`. (10p)

2. Show the parse tree and the abstract syntax tree of the statement

```
while (x++) if (cond) ++x ; else x++ ;
```

in the grammar that you wrote in question 1. (5p for each tree)

3. Write syntax-directed type inference code (or pseudocode) for pre- and post-increments, that is, expressions of forms `++x` and `x++`, where `x` is a variable of type `int` or `double`. (5p)

Also write syntax-directed interpreter code (or pseudocode) for pre- and postincrements as specified above. (5p)

Group 2: intermediate questions

4. Show a regular expression and a finite-state automaton recognizing sequences of `a` and `b` of at least two symbols, such that the second-last symbol is `a`. Examples: `aa`, `aab`, `bbbbab`. (3p for the expression, 2p for the automaton if nondeterministic, 7p if deterministic)

5. Write a compilation scheme for `while` statements in JVM (i.e. Jasmin assembler), and explain each of the instructions occurring in it by giving its small-step semantics. It is not necessary to remember exactly the names of the instructions - only what arguments they take and how they work. (4p for the scheme, 6p for the semantics)

Group 3: advanced questions

6. Define the Church numerals 0..5 in pure lambda calculus (3p). Then define the addition of Church numerals (4p) and show, in around 5 steps, the computation of 3 + 2 (3p).