Programming Languages

Example Exam, 7 March 2008

February 24, 2011

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 at least half of the intermediate is needed to get mark 4. In addition to all the easy and the intermediate, about one third of the advanced is 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 require any material from the lecture notes. The advanced questions may also require material from the book, chapters 1-6.

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 kinds of constructs in C++:

   - declarations of one variable, as `int x ;`
   - `while` loops
   - `if` statements with `else`
   - statements formed from expressions by adding a semicolon `;`
   - expressions that are identifiers or integer literals
   - the types `int` and `double`.

   An example statement is shown in question 3. 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 (foo)
  if (cond) int x ; else double y ;
```

in the grammar that you wrote in question 2. (10p)

3. Write a syntax-directed type inference rule of expressions of the form `exp1 + exp2`, where `+` is overloaded and the operands can be either `int` or `double`, and no type casts are permitted. This rule has to return a type. (5p)

Write syntax-directed interpretation rules for pre- and post-increment expressions of forms `++x` and `x++`, where `x` is a variable of type `int` or `double`. (5p)

Group 2: intermediate questions

4. Show a regular expression and a deterministic finite automaton recognizing sequences of `a` and `b` of at least two symbols, such that the second-last symbol is `a`. Examples: `aa`, `aab`, `bbbbab`. (10p)

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. (10p)

Group 3: advanced questions

6. Write operational semantic rules showing the difference between call-by-value and call-by-name lambda calculus. Also give an example of an expression that behaves differently in these two kinds of semantics. (10p)