Interpretation and Compilation
EXAM
14 January 2015 (09:00-12:00 – duration 3 hours)

NOTE: The solution to the exam is to be submitted in a fully written form, as a standard exam (not as program code). However, you can experiment with your code and use your laptop or PC to code parts of the exam and test them, if you want.

Consider the programming language defined by the following abstract syntax:

exp ::= exp + exp exp - exp exp * exp exp / exp
exp == exp exp ? exp exp num id
decl id (id*)=e **in** e **end** f(e*)

This is a very simple language with (named) function declarations, the basic types are just integers and booleans. The declared functions cannot be recursive. Notice that the defined function can have zero or more parameters, and, to simplify the exercise, only a single function can be declared each time by the **decl** construct.
Function parameters and function results can only be of basic types (integers or boolean), also to keep things manageable. The only possible usage occurrence of a function identifier F in a well-typed program is in a function application F(e*). An example of a source program is the following:

decl f(x) = (x==0)?1:(1000/x) **in** f(10) **end**

which evaluates to the integer result 100.

QUESTON 1. Using the notation adopted in our course lectures, write the typing rules for the following constructs:

- **decl** id (id*)=e **in** e **end**
- f(e*)

Assume the java based abstract syntax representation studied in the course, using a separate java class for each AST node type.
Recall that every AST node type must comply with the interface

```
interface ASTNode {
IValue evaluate(Environ<IValue> e);
IType typecheck(Environ<IType> e);
}
```

We denote by Environ<X> is the standard java implementation of environment used in the course, which binds identifier names to entities of type X. The type X represents program values in the case of the evaluation environment Environ<IValue> and types in the case of the typecheking environment Environ<IType>. But you can use other (good) environment implementation of your preference, if you want.

QUESTION 2. Write the java implementation of the AST node classes for the following two constructs

decl id (id*)=e **in** e **end**

f(e*)

In particular detail the evaluate method (which, besides evaluation must also perform full dynamic type checking) and the typecheck method (which must perform full static type checking).

Now, the final part of the exam relates to compilation. We will use the JVM as target architecture.

QUESTION 3: Consider again the following two key constructs

decl id (id*)=e **in** e **end**

f(e*)

Define the compilation scheme for each of these two constructs. Use the notation

[[e]]E

where [[e]]E denotes the code generated for expression e under environment E.

To define the compilation schemes, you can assume given a function findl(E,id) that returns the offset of identifier id in environment E. That is, if identifier id is in the topmost level in E then findl(E,id) returns 0, if id is in the next level above the top one, then findl(E,id) returns 1, etc.

QUESTION 4: Sketch the code generated for the sample program

decl f(x) = (x==0)?1:(1000/x) **in** f(10) **end**