Interpretação e Compilação de Linguagens– 2016-2017
Interpretation and Compilation of Programming Languages

Final Test December, 14, 2016

Author: João Costa Seco
Notes: The test is open book. Students can use any (individual) printed material that each student brings along. The test has a duration of 1h30.

Q-1 [10 val.] This question is about the definition of an abstract syntax and the operational semantics for a functional programming language with only pairs and pattern matching. Consider the programming language, called `OnlyPairs`, presented in class with the concrete syntax given by the following grammar:

$$E ::= num \mid E_1{+}E_2 \mid (E_1, E_2) \mid x \mid \mathsf{match}\ (x, y) = E_1\ \mathsf{in}\ E_2 \mid \mathsf{fun}\ x \Rightarrow E \mid E_1(E_2)$$

The language comprises the base constructs for: **integer literals** (num), and their usual operations, represented here by operation $E + E$; It also includes the constructor of **pair values** ((E, E)), **identifier** use (x) and the pair deconstructor, by **pattern matching** $\mathsf{match}\ (x, y) = E_1\ \mathsf{in}\ E_2$, where x and y are bound to the values of the pair denoted by expression E_1 and whose scope is expression E_2. The language also includes **functions** as first-class values, their constructor ($\mathsf{fun}\ x \Rightarrow E$) and the corresponding function call expression ($E_1(E_2)$).
Consider the example written in the programming language `OnlyPairs`:

```
match (f,w) = ((match (x,y) = (1,2) in fun z => x + y + z), 3) in f(w)
```

a) [**1 val.**] **Define** the abstract syntax of the **match operation** in language `OnlyPairs` by means of an abstract data type, defined by set of (abbreviated) Java classes and interfaces.

b) [**1 val.**] **Define** the set of values of language `OnlyPairs` by means of an abstract data type, defined by a set of (abbreviated) Java classes and interfaces.

c) [**3 val.**] **Define** the operational semantics for the **match expression** in the language `OnlyPairs` by means of a method **eval**.

d) [**1 val.**] **State** the denotation (value) of the example above according to the semantics defined in the previous question, and the expected semantics for the remaining operators.

e) [**2 val.**] **Show** that it is possible to encode the standard identifier declaration ($\mathsf{decl}\ x = E\ \mathsf{in}\ E$) using the constructions of language `OnlyPairs`.

f) [**2 val.**] **Show** that it is possible to encode the standard pair operations ($\mathsf{fst}\ E$ and $\mathsf{snd}\ E$) using the constructions of language `OnlyPairs`.

Q-2 [**7 val.**] Consider the language `AlsoRecords`, obtained by extension of language `OnlyPairs`, with a record constructor and a field selection operation.

$$E ::= \ldots \mid \{\ell_1 = E_1, \ell_2 = E_2, \ldots\} \mid E.\ell$$

This question is about the definition of the type system for language `AlsoRecords`. To answer the following questions you may use abstract data types, defined by a set of Java classes and interfaces, and the corresponding methods using Java Code. You may also use the notation used in the lecture slides.

a) [**2 val.**] **Define** the set of types used to type programs of language `AlsoRecords`.

b) [**3 val.**] **Define** the type system of language `OnlyPairs` by means of a **typecheck** function, for the **match expression**, the **record construction** and **field selection** expressions.

c) [**1 val.**] **Enumerate** the execution errors that may occur during the execution of a program written in language `AlsoRecords`, according to the semantics defined in question **Q-1**, and the corresponding semantics for records.

d) [**1 val.**] **Indicate and justify** which execution errors may be prevented by the type system, and those that cannot.

Q-3 [**3 val.**] This question is about the compilation of programs in langauge `OnlyPairs`. Consider the following program written in the `OnlyPairs` language.

```
match (x,y) = (2,3) in match (z,w) = (x*y,x+y) in z+w
```

a) [**1 val.**] **Indicate** what supporting Jasmin classes would be needed in a type preserving compilation procedure for the example above.

b) [**2 val.**] **List** the set of instructions that results from translating the expression above to Jasmin assembly code.