INSTRUCTIONS FOR THE ONLINE MIDTERM ICL EVALUATION

Dear all,

23 November 2020 (11:00), you will start the midterm evaluation session.

A zoom session will be opened at 11:00 (usual ICL 2020 link).

After that, I will open a (google) form that you will need to fill and submit.

The web link is here (clickable)

https://docs.google.com/forms/d/e/1FAIpQLSeL1uUUPybsNo0cN9b31F651b7I1Qp4nXgYNZ92qxBGypL42A/viewform

The form only contains the fields for you to write down the answers, not the questions. There are four questions (some with several items)

The questions will be available in a PDF file uploaded to the CLIP before the test starts.

To access the form you will need to authenticate with
your official student FCT NOVA email, which will be recorded in the form.

---@campus.fct.unl.pt

The evaluation will take place 11:15-13:00.

You may edit and re-submit your form several times until the end of the test.

During the evaluation, I will be answering questions on the zoom chat.

Thanks, all the best

Luis Caires

Interpretation and Compilation / MIEI / FCT UNL

ONLINE Midterm Test 2020

ZERO. The abstract syntax tree for expression 2+3 may be constructed in Java by

new ASTAdd(**new ASTNum**(2), **new ASTNum**(3))

using our standard AST model. In the same way, indicate Java code for constructing abstract syntax trees for the expressions:

(a) 2*3+1*2
(b) –(2+(4+3))
(c) (2+4)*3

ONE. Consider the basic language we have studied in the course, and the programs. For each program, indicate the result of its evaluation and the state of the environment at the point where the boxed expressions start to be evaluated.

```
def x=1 y=2 in
  def z = x+y w = x-y in
    z + w
  end
end
```

```
def  x = 10 in
  (def x = 1 in x + x end) * (def y = 1 in x + y end)
end
```

NOTE: To draw the state of an environment use a notation like

[a-> 0, b -> 4, c -> 6]

[x-> 5, z -> 9]

[i -> 0] (<= env top level)

Here we have represented an environment with three levels, in which identifier i is declared in the level at the top level (top of "stack")

TWO. Consider the following expression language based on the basic language we have studied in the course, but where now we have "structured identifiers"

SId ::= **id | up** SId
E :: = **num** | SId | E **+** E | E **–** E | E ***** E | **def** (Id **=** E)+ **in** E **end**

Structured identifiers have the form **up** … **up** Id where Id is a simple identifier as in the basic language, and **up** is a unary operator.

An example of a program is

```
def x=2 y=1 in
  def z = x+y x = x - y in
    def y =  z +x in
       y + up y
    end
  end
end
```

This program evaluates to **6**, which is the value of the body y + **up** y. In this expression x + **up** y, while y evaluates to **5**, (**up** y) evaluates to **1**, which is the value of y in the first enclosing outer scope. In general, a sequence of *K* **up**'s prefixing a simple identifier X denotes the value of X in the *K* -th outer scope.

Explain how you would extend your interpreter in order to add the **up** operator to the language with expressions. Explain **briefly but clearly**

1. additions to the language grammar,
2. additions to the abstract syntax, in particular write the class **ASTUp**.
3. define the eval method for the class **ASTUp**
4. any modifications to the environment class, if needed.

THREE. Consider the compiler for the language with definitions.

(a) Indicate all the JVM code generated for the program

```
def y = 1 in
  def x = 2 in
     x + y
 end
end
```

(b) Consider a compiler to the language in **TWO**, which only involves the definition of a method **compile** to the class **ASTUp.**

Indicate the JVM code that should be generated for the following program (in the main you only need to indicate the code between START and END)

```
def z = 1 in
  def z = 2 in
     z + up z
 end
end
```

(c) Write the code for the **compile** method in class **ASTUp**, with signature

void compile(CodeBlock c, Env e)