# *ADVANCED PROGRAMMING*

The AP Programming Project for the February exam must be submitted by e-mail by **15-02-2015 (midnight)** to **giangi@di.unipi.it** and **cisterni@di.unipi.it** with the subject prefix **[AP-Project]**.
You can solve exercises using a programming language of choice among: C/C++, Java, C#, F#, Javascript. The submission must include all required files for compiling and executing the proposed solution.
It is allowed to develop the solution into teams however the proposed solution should be discussed individually.

## 1. The TinyJs Programming Language

TinyJs contains a small set of data types. It has the three primitive types *boolean*, *number*, and *string* and the special valued *undefined*. You can assume standard operators for the TinyJs data types.

As usual, variables are containers for storing data values. In this example, x, y, z and w, are variables created with the **var** keyword

```
var x = 5;
var y = 6;
var z = x + y;
var w;
```

The variable w declared without a value will have the value **undefined**.

Functions look like C functions, except that they are declared with the **function** keyword instead of a type. When calling a function, it is not required that you pass a fixed number of parameters. Excess parameters are ignored. Missing parameters are given the value undefined. Functions have lexical scoping and are first order values

The set of statements includes var, if, switch, for, while, do, break, continue, return. They work the same as in other programming languages.

TinyJs includes the **`eval()`** function. The **`eval()`** function evaluates TinyJs code represented as a string. Few examples follow:

```
var x = 2;
var y = 39;
var z = "42";
eval("x + y + 1"); // returns 42
eval(z);           // returns 42

var str = "if ( a ) { 1+1; } else { 1+2; }";
var a = true;
var b = eval(str);  // returns 2
```

```
var exp = "var x=5;var y=6;function mult(a){return a * y;}";
eval(exp);
var c = mult(x) + 4;
println(c);    //return 15
```

```
var strfun = "function myfun(arg) { return arg * 3; }";
eval(strfun);
var function usefun(x) {   var res = myfun(x); return res; }
var myres = usefun(5)  //return 8
```

## 2. Implementing TinyJs (Mandatory)

Define the parser and the interpreter for the TinyJs language.
It is mandatory to implement the type checking required to ensure that TinyJs programs operate correctly with types.
It is up to the candidate to fill the gaps in the specification and provide motivations for the choices made.

## 3. Extending TinyJs (Optional)

The extension requires modifying the language to distribute computation over a network.

The special function called "**distribute_eval()**" takes care of distributing the computation. This primitive function takes a network address (a URL) and a program represented as a string. Here is an example.

```
var exp = "var x=5;var y=6;function mult(a){return a * y;}";
distributed_eval(exp, u);
var c = distributed_eval("mult(x) + 4");
println(c);    //return 15
```

Define the parser and the interpreter for the TinyJs language extended with **distribute_eval().** It is up to the candidate to fill the gaps in the specification and provide motivations for the choices made.