Formal Languages and Compilers
Proff. Breveglieri, Morzenti
Written exam[1]: laboratory question
11/02/2014

SURNAME: .....
NAME: ..... Student ID: .....
Course: ○ Laurea Specialistica ○ V. O. ○ Laurea Triennale ○ Other: ...
Instructor: ○ Prof. Breveglieri ○ Prof Morzenti

The laboratory question must be answered taking into account the implementation of the `Acse` compiler given with the exam text.

Modify the specification of the lexical analyser (`flex` input) and the syntactic analyser (`bison` input) and any other source file required to extend the `Lance` language with the ability to handle conditional expressions on syntactically similar to the ones of the common Python programming language. An example of such a conditional expression follows:

```
int a;
int b;
read(a);
read(b);
b = a * 2 - b if a > 2 * b else 0;
write(b);
```

The conditional expression is defined for any value of the condition indicated between the `if` and `else` keywords: in case the condition is true (i.e. non zero) the value of the conditional expression is the result of the expression preceding the `if` keyword. If the condition is false, the conditional expression evaluates to the result of the expression after the `else` keyword. Take care to specify the precedence and associativity of the operators in such a way to allow the composition of conditional expressions.

Implementations of the construct with a linear execution flow will be *preferred*.

[1] Time 60'. Textbooks and notes can be used.
Pencil writing is allowed. Write your name on any additional sheet.

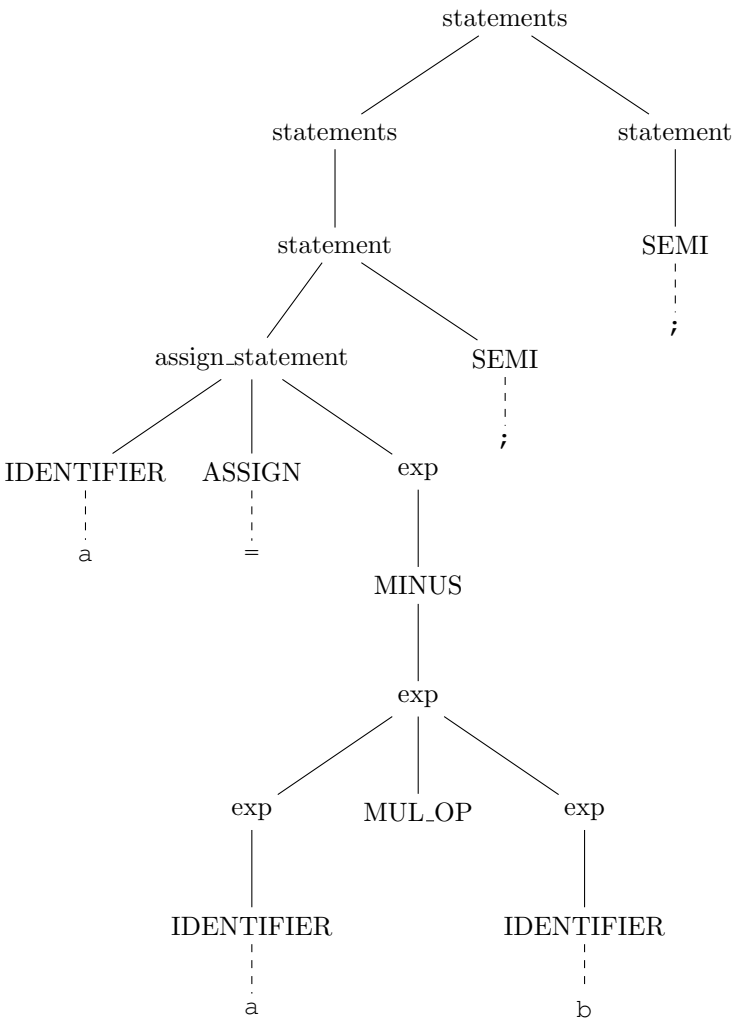
1. Define the tokens (and the related declarations in **Acse.lex** and **Acse.y**). (2 points)
2. Define the syntactic rules or the modifications required to the existing ones. (3 points)
3. Define the semantic actions needed to implement the required functionality. (20 points)

   The solution is in the attached patch.

4. Given the following `Lance` code snippet:

```
a = -a * b;;
```

write down the syntactic tree generated during the parsing with the Bison grammar described in Acse.y *starting from the* `statements` *nonterminal.* (5 points)



5. (**Bonus**) Describe how the implementation of the conditional expression construct should be extended to allow the specification of a value for the expression, should the result evaluate to zero after the conditional evaluation. Such a value is specified after the `onzero` keyword as follows:

```
x = a if c > 2 else b onzero 42;
```