Formal Languages and Compilers
Proff. Breveglieri, Morzenti
Written exam[1]: laboratory question
03/09/2014

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 the arithmetic `if` construct. Consider the following code snippet as an **illustrative example** of its use:

```
int a,b;

over: a=b;
if A( a-b )A sum,over,print;
b=b*2;
print: write(a);
sum: a=a+b;
```

The arithmetic `if` construct evaluates the expression enclosed by the special parentheses `A(` and `)A` and, depending on whether its result is negative, zero, or positive, causes the control flow of the program to continue from the first, second, or third label which follow it. The syntax for the labels and labelled statements is the same as the one of `C` language. At most one label per statement is allowed. If the arithmetic `if` construct points to a label which is not present in the program, the modified compiler should raise a *compile time* error

[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)

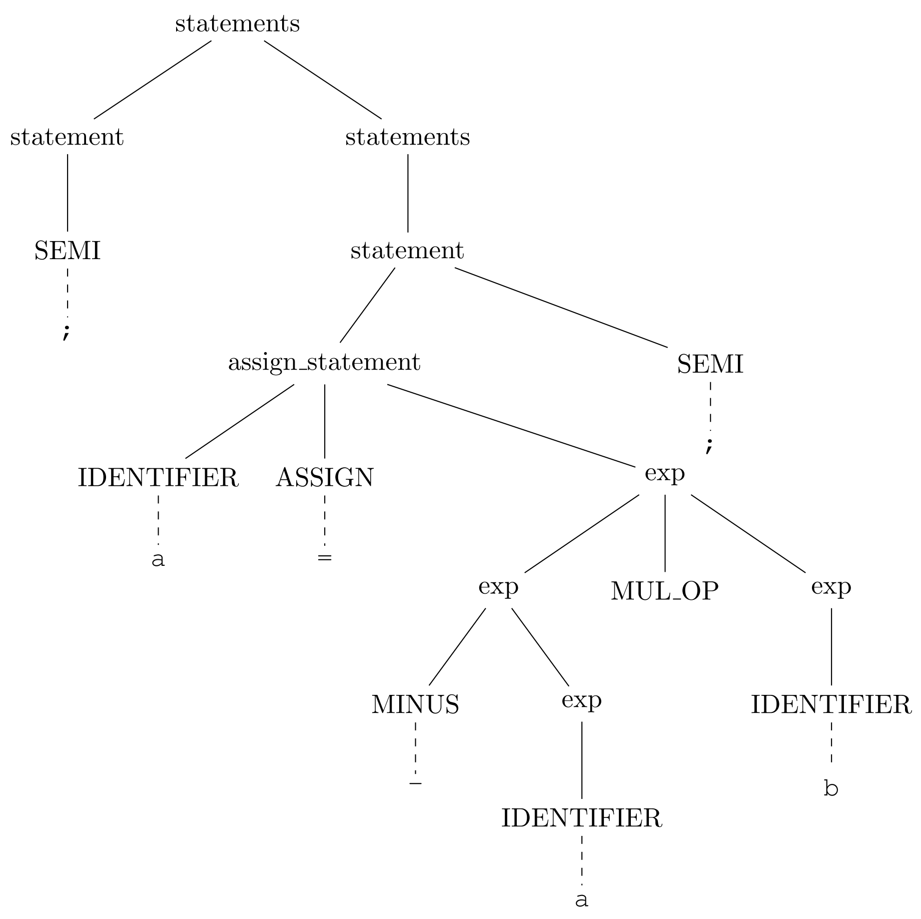
3. Define the semantic actions needed to implement the arithmetic `if` statement. (20 points)

   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**) A common misuse in the languages supporting the arithmetic `if` is to employ it to build iterative constructs. Describe how it is possible to detect such a behaviour at compile time and emit a warning for the programmer.

   It is possible to detect if there is a misuse of the `if` construct through checking whether the labels are fixed in such a way that at least one of them is an actual backedge. This can be done quite easily as in syntax directed translation, it is sufficient to check that all the labels are already fixed when parsing the if construct. To point this out to the programmer, it is sufficient to print a warning message at compile time.