Formal Languages and Compilers
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Written exam[1]: laboratory question
27/06/2009

SURNAME: .....
NAME: ..... Student ID: .....
Course: ○ Laurea Specialistica ○ V. O. ○ Laurea Triennale ○ Other:.....
Instructor: ○ Prof. Breveglieri ○ Prof. Crespi ○ 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 analyzer (`flex` input) and the syntactic analyzer (`bison` input) and any other source file required to extend the `Lance` language with the ability to *use assignments as a part of an expression* with the following syntax (the same as C programming language):

```
int x, y, z, a[3];

x = y = z = a[0] = 1;
write( x = y + 3 );
z = (y = 2) * x;
x = z + y;
write( x );
```

Compiling this code snippet and executing it would result in a printout of the values "4" and "10".

The solution must comply with the following specifications:

- The assignment operator has the lowest priority and is *right* associative.
- An assignment expression may appear in the grammar everywhere a nonterminal expression does (i.e., everywhere a nonterminal `exp` is placed).
- The left hand side of an assignment may be either a variable or an element of an array.
- The right hand side of an assignment may be any valid expression of the language, including an assignment.

Your modifications have to allow the `Acse` compiler to both correctly analyze the syntactical correctness of the aforementioned constructs and to generate a correct translation in the `Mace` assembly language.

Take care not to build an ambiguous grammar.

[1] Time 45'. Textbooks and notes can be used.
Pencil writing is allowed. Write your name on any additional sheet.

1. Define the syntactic rules needed (or the modifications to the existing ones) in order to obtain the required functionalities, including possible directives for precedence and associativity. (7 points)

2. Define the semantic actions needed (or the modifications to the existing ones) in order to obtain the required functionalities. (8 points)

3. Given the following code snippet:

```
int a = 2;
int b;

b = a*5 + (a = a - 1) + a*2;
write( b )
```

   (a) Write down the syntactic tree of the assignment statement on the third line, starting from the non-terminal `statements` according to the grammar specified before. (4 points)

   (b) According to the grammar specified before, write down the evaluation order of the operations performed in the aforementioned statement and compute the final value of the variable `b`. (5 points)

4. In the code snippet reported in the question 3 the value of `b` depends on the order in which the operands are evaluated. Since a typical programming language doesn't suffer from such issues, delineate which semantic check may be applied by the compiler in order to produce a warning in situations like the ones in the example. It is not required to write the actual code, but the semantic actions and the structures involved must be specified correctly. (9 points)