Formal Languages and Compilers
Proff. Breveglieri, Morzenti
Written exam[1]: laboratory question
04/02/2016

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 **sum-out of-as** construct.
This construct allows to iterate the computation of the expression specified after the `as` keyword over pairs of contiguous elements of a given array. The name of the array is specified after the `out of` keywords, while the name taken by the variables in the expression to be computed is specified right after the `sum` keyword. The results should be accumulated via addition in a scalar variable placed as the left hand side of the assign statement, of which the **sum-out of-as** construct constitutes the right hand side.

An example is provided in the following code snippet

```
int i,j,r,v[4];
v[0]=1;
v[1]=4;
v[2]=5;
v[3]=6;
r = sum i,j out of v as i+j*2;
/* r = 1+4*2 + 4+5*2 + 5+6*2 */
r = sum i,j out of v as 3*2-1;
/* r = 5 + 5 + 5 */
```

In case the variable specified after the `out of` keywords is not an array, the modified compiler should report a compile-time error, in any fashion preferred by the implementor. The semantic of the construct does not specify the order in which the elements of the array should be evaluated.

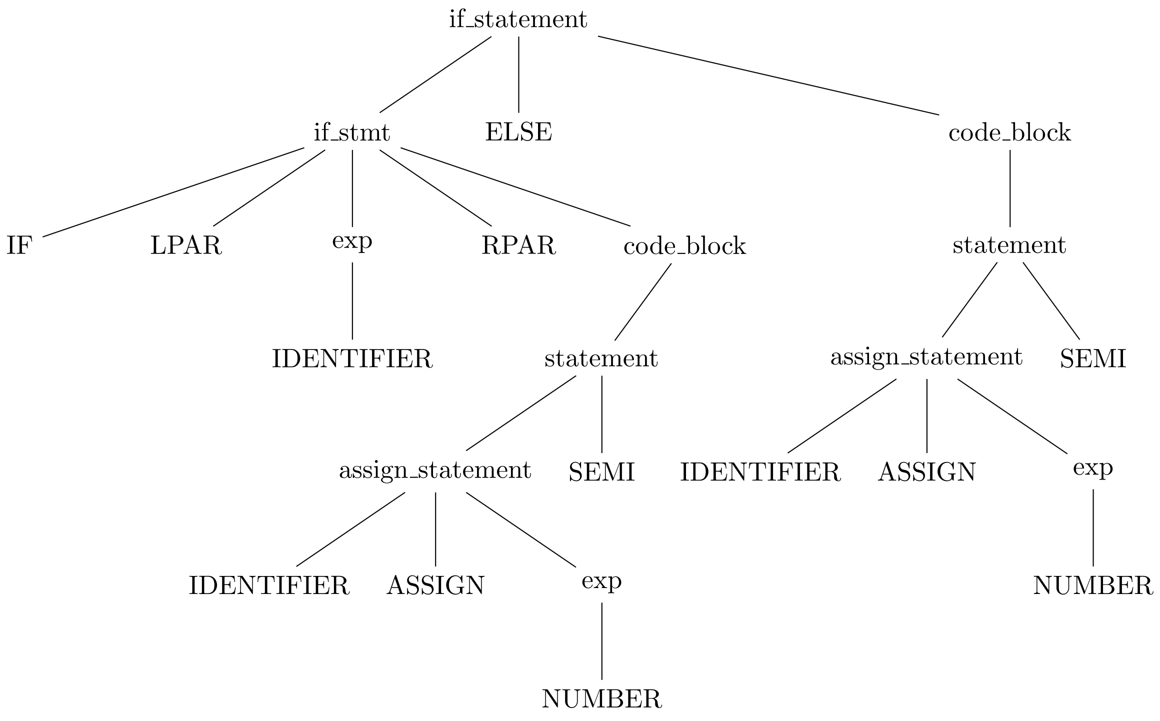
[1]Time 60'. Textbooks and notes can be used.
Pencil writing is allowed. Write your name on any additional sheet.

1. Define the tokens (and the related declarations in **Acse.lex** and **Acse.y**). (3 points)
2. Define the syntactic rules or the modifications required to the existing ones. (4 points)
3. Define the semantic actions needed to implement the required functionality. (18 points)
   The solution is in the attached patch.

4. Given the following `Lance` code snippet:

```
if (a)
  b=1;
else
  a=1;
```

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 to modify your solution to extend the aforementioned construct to allow a sequence of elements of arbitrary length to be employed in the expression, specified as a comma separated list in place of the current pair of identifiers.