Formal Languages and Compilers
Proff. Breveglieri, Morzenti
Written exam[1]: laboratory question
18/06/2018

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 new **exists** construct.

The new construct implements the first-order existential quantification over the positions of the array appearing in the *expression* between parenthesis. The identifier used to access the array positions is defined after the keyword exists. Possible clashes are handled according to the semantics of the first-order logic: the inner-most identifier is considered for accessing the array. No nested exists are allowed, hence the implementation should provide a proper syntactical check. The following restriction is considered: only one array identifier occurs in the expression of any exists construct (hence, no checks has to be implemented).
An example is provided in the following.

```
int a[5];
int x = 3;

a[0] = 0; a[1] = 1; a[2] = 2; a[3] = 3; a[4] = 4;

if ( exists i (a[i]>2) ) write(1);
else                     write(0);
// it prints 1

if ( exists i (a[x]>2) ) write(1);
else                     write(0);
// it prints 1

if ( exists x (a[x]>10) ) write(1);
else                      write(0);
// it prints 0
```

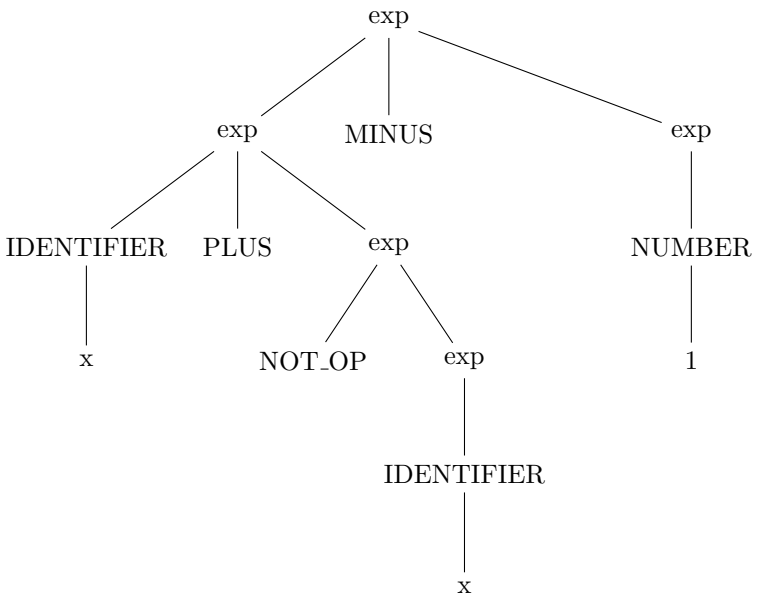
[1] Time 60'. Textbooks and notes can be used.
Pencil writing is allowed. Write your name on any additional sheet.

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)

4. Given the following `Lance` code snippet:

```
x + !x - 1
```

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



5. (**Bonus**) Describe how to modify the implementation given at point 3 in order to allow an existential quantification over more than one index variable like, for instance, `exists i,j,k (a[i]>0 && b[j]==1 && c[k]==2)`.

多个 index，如 i，j，k