Formal Languages and Compilers
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Written exam[1]: laboratory question
25/09/2013

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 analyser (`flex` input) and the syntactic analyser (`bison` input) and any other source file required to extend the `Lance` language with the statement *loop-decreasing*.

```
int x[100], input, c = 0, s = 100;

loop_decreasing s by c  {
  read(input);
  c = c + 5;
  write(c);
} while (input > 32 && c < 50);

write(input);
write(s);
```

Figura 1: Example

This kind of **loop** takes as parameters the *counter variable*, the *decrement expression* and an *execution condition*. The construct has the following constraints:

- a **positive** loop counter variable is a necessary condition for the loop body execution,
- the execution of all the iterations **except for the first** is controlled also by the *execution condition*: if the condition is false the control flow exits the loop. In other words, the execution condition is evaluated at the end of the loop body.

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

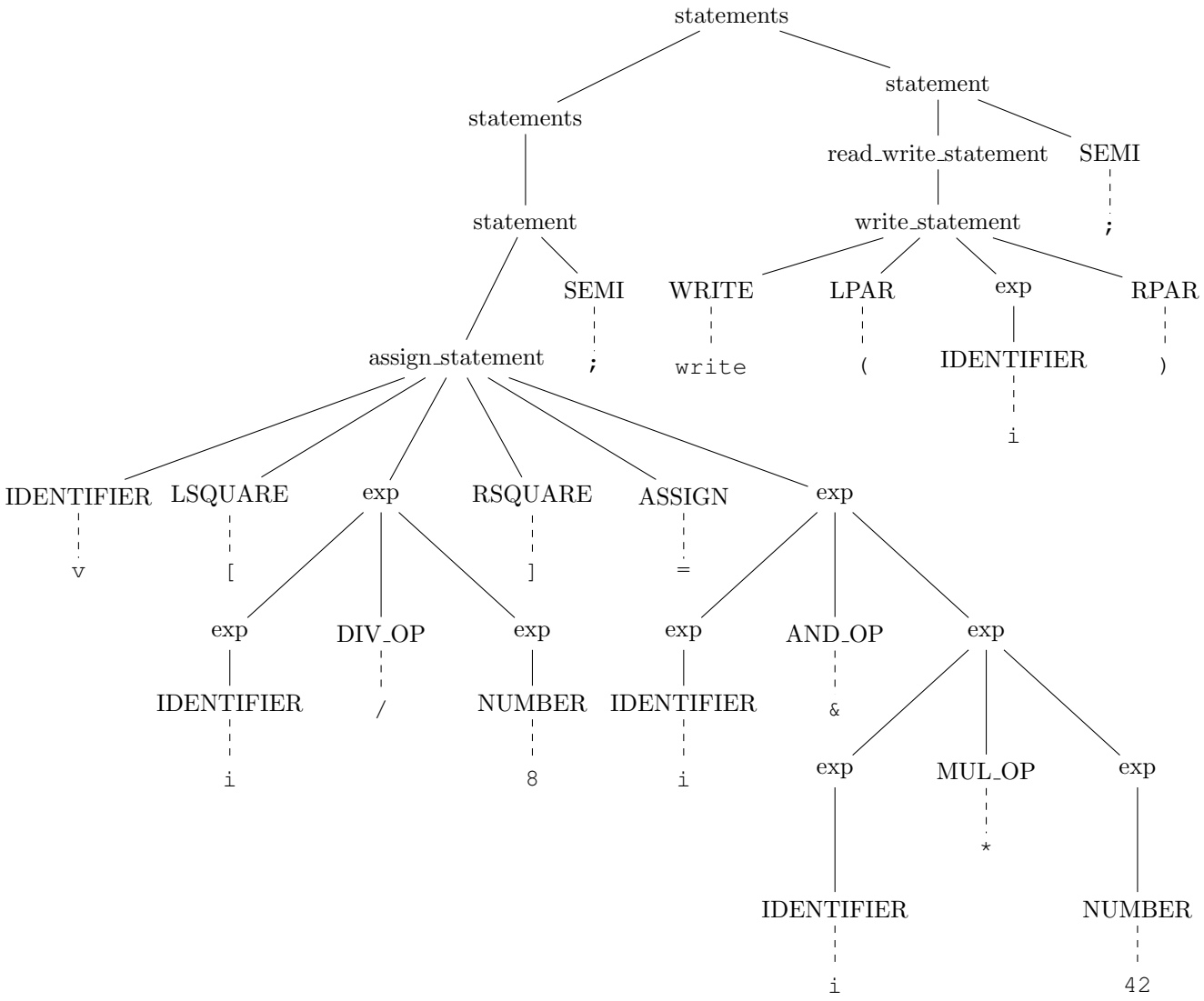
1. Define the tokens (and the related declarations in `Acse.lex` and `Acse.y`). (2 points)
2. Define the syntactic rules or the modifications required to the existing ones. (4 points)
3. Define the semantic actions needed to implement the required functionality. (18 points)

   The solution is in the attached patch.

4. Given the following `Lance` code snippet:

```
int v[100], i;
v[i / 8] = i & i * 42;
write(i);
```

write down the syntactic tree generated during the parsing with the Bison grammar described in Acse.y *starting from the* `statements` *nonterminal.* (6 points)



5. (**Bonus**) Describe how you can extend the *loop-decreasing* construct adding a variant that takes the initial counter as an expression instead of a variable.
   Provide the modified `bison` grammar that shows *both variants* of loop-decreasing construct.
   A full implementation is optional.

```
int x,y,i;
read(x);
read(y);
loop_decreasing
  from x + 100 by 14 - y {
  i = i + 1;
  write(i);
} while (y + i < 100);
```