Formal Languages and Compilers
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Written exam[1]: laboratory question
17/09/2014

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 ability to implement the **cond** instruction. The following Figure 1 shows an example. The instruction gathers a non-empty list of conditional cases that are, possibly,

```
int x, y;

read(x);

cond{
  case x>0: read(y);
  case x+1: x=0; y=0;
  default: y=x;
}

write(x);
write(y);
```

Figura 1: Example

terminated by a **default** case. Each conditional case is identified through a token **case** followed by an expression, a symbol : and a non-empty list of instructions. A conditional case is executed when the associated expression is verified (i.e. its value is non-zero) and it is the first conditional case of the list whose expression is verified (*meet-first*). In other words, this occurs when all the previous expressions associated with the cases preceeding the one that is executed, are not met. The default case, if defined, is executed if none of the previous conditional cases in the body is performed. REMARK: After the execution

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

of a conditional case, the execution flow jumps to the instruction following the **cond** instruction.

For example, if $x = 1$ after instruction $read(x)$, then $read(y)$ is performed and, finally, the two *write* instructions. If $x = 0$ after instruction $read(x)$, the sequence $x = 0; y = 0;$ is performed and, finally, the two *write* instructions. If $x = -1$ after instruction $read(x)$, the default case is performed.

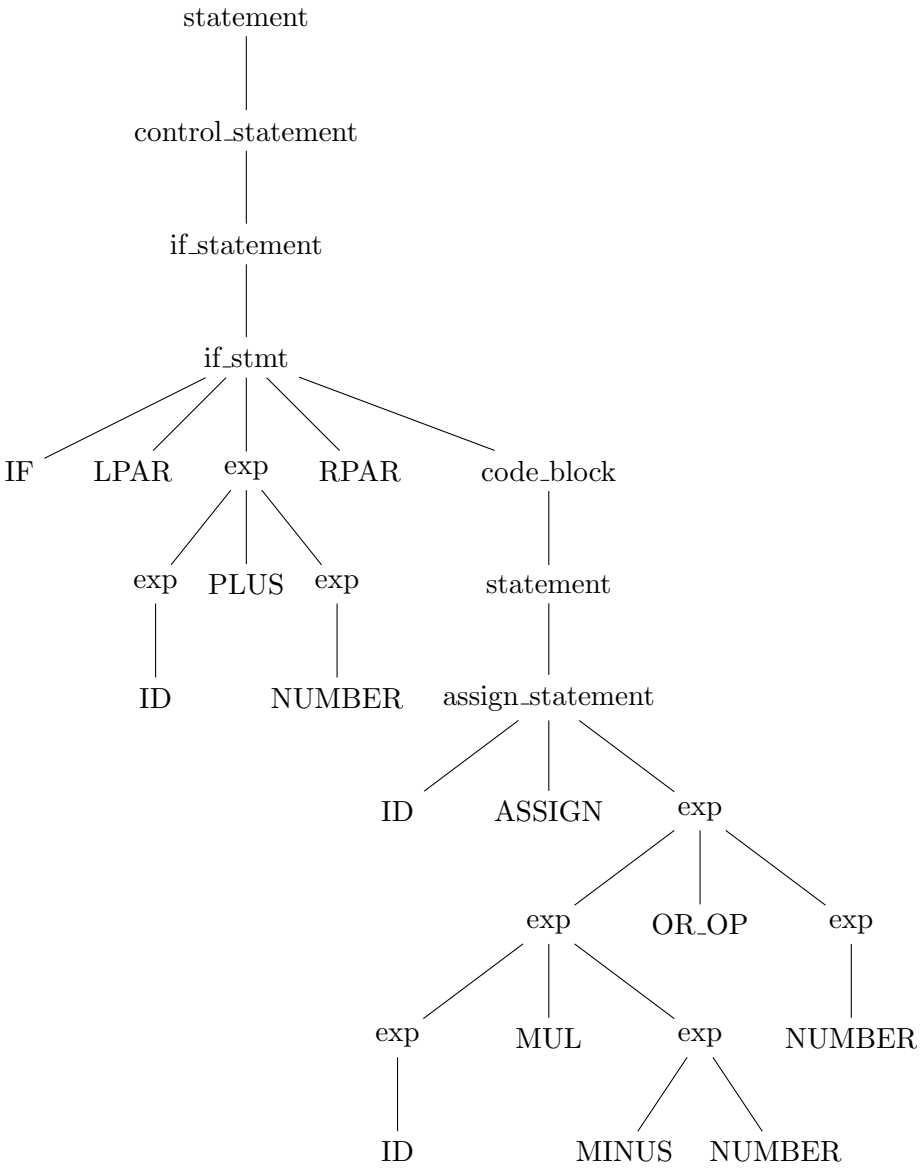
1. Define the tokens (and the related declarations in **Acse.lex** and **Acse.y**). (3 points)
2. Define the syntactic rules or the modifications required to the existing ones. (4 points)
3. Define the semantic actions needed to implement the required functionality. (18 points)

   The solution is in the attached patch.

4. Given the following `Lance` code snippet:

```
if(x+1)
  x=y*-2|1;
```

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



5. (**Bonus**) Explain how to modify **cond** instruction so that conditional cases with empty list of statements are allowed. In this case, the semantics is such that the expressions of **case** are considered as they are written through logical disjunction (in other words, an OR operation is inherently applied). For example, in Figure 2 the instruction $x = 0; y = 0;$ are performed if $x > 0 \vee x + 1 \neq 0$.

```
int x, y;

read(x);

cond{
  case x>0:
  case x+1: x=0; y=0;
  default: y=x;
}

write(x);
write(y);
```

Figura 2: Example

The assignment requires a new syntactic rule which defines the conditional cases with empty list. The associated semantic action adds a branch (**bne**) instruction after the exp expression in the current conditional case (with empty list) that brings the execution flow to the statement that is defined into the first conditional case with non-empty list following the current one. The label used in the branch instruction is stored into the structure associated with the current **cond** instruction.