Formal Languages and Compilers
Proff. Breveglieri and Morzenti
Written exam[1]: laboratory question
05/07/2017

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 domain of expressions of `Lance` language with the representation of positive and negative *infinite* values and the *undefined* value. **At runtime**, each expression is represented by means of a *value* and a descriptor *inf*: "value" is either the integer value of the expression, when the expression is defined and different from an infinite, or the sign of the infinite (+/−). Value "inf" specifies if the expression is an integer or an infinite or if it is an undefined value. Two new constants are introduced in the language to represent infinite values (i.e., **+infty** and **-infty**) and one is added to represent the undefined value (**NaN**). The following example shows the use of infinite values in case of assignment instructions and conditional expressions of the form $exp == exp$, the latter being defined according to the following semantics:

- if an undefined is compared with an expression the resulting expression is **undefined**.
- Otherwise, the resulting expression is **defined** and the *value* of the expression is established according to following cases, where v is an integer value and s in front of symbol ∞ is the sign of the infinite in $\{+, -\}$:

exp_1	exp_2	$exp_1 == exp_2$
$\pm\infty$	v	*false*
v	$\pm\infty$	*false*
$s_1\infty$	$s_2\infty$	$s_1 == s_2$
v_1	v_2	$v_1 == v_2$

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

```
int a,b,c,d;

read(a);
read(b);                  // c, d are undefined

b = +infty;
c = -infty;               // d is undefined

write(b == b);             // true
write(a == b);             // false
write(b == c);             // false
write(d == c);             // false

write(+infty == -infty); // false
write(+infty == +infty); // true
write(-infty == -infty); // true
write(Nan == Nan);         // false
write(Nan == b);           // false
```

1. Define the tokens (and the related declarations in **Acse.lex** and **Acse.y**). (1 points)
2. Modify the struct `t_axe_expression` to allow the description of infinite and undefined expressions. (2 points)
3. Define the syntactic rules or the modifications required to the existing ones, if they are needed. (2 points)
4. Modify the semantic actions required to handle value assignments. (5 points)
5. Modify the semantic action associated with equality operator == in the following two cases **only**: when two immediate are compared and when two non immediate expressions are compared each other. (14 points)
6. The current version of the compiler only deals with the value of a variable. Modify the compiler in order to properly initialize and preserve runtime information related to the infiniteness/undenedness of a variable properly updated along the execution of the code. (8 points)