Formal Languages and Compilers
Proff. Breveglieri and Morzenti
Written exam[1]: laboratory question
02/02/2017

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 `Lance` language with the operator **(bit)**. The operator can be written in front of a *variable identifier*. It transforms a vector v of integers into an integer, according to the following semantics: if $v[i]$ is 0 then the i-th corresponding bit of $(bit)v$ is 0, otherwise if $v[i]$ is *non* 0 then the i-th corresponding bit is 1. In any case, at most 32 elements of the vector are considered (**int** variables are defined over 32 bits). Note that, if **(bit)** is applied in front of an identifier of a scalar, the returned value is the same as the one stored in the variable (namely, no translation is carried out).

```
int v1[3];
int v2[1];
int v3[40];
int x = 10;

//v1 encodes 110
v1[2] = 15; v1[1] = 79; v1[0] = 0;

//v2 encodes 1
v2[0] = 1;

//v3 encodes -2^(31)
v3[31]=37;

write((bit)v1); //output 6

write((bit)v2); //output 1

write((bit)v3); //output -2^(31)

write((bit)x);  //output 10
```

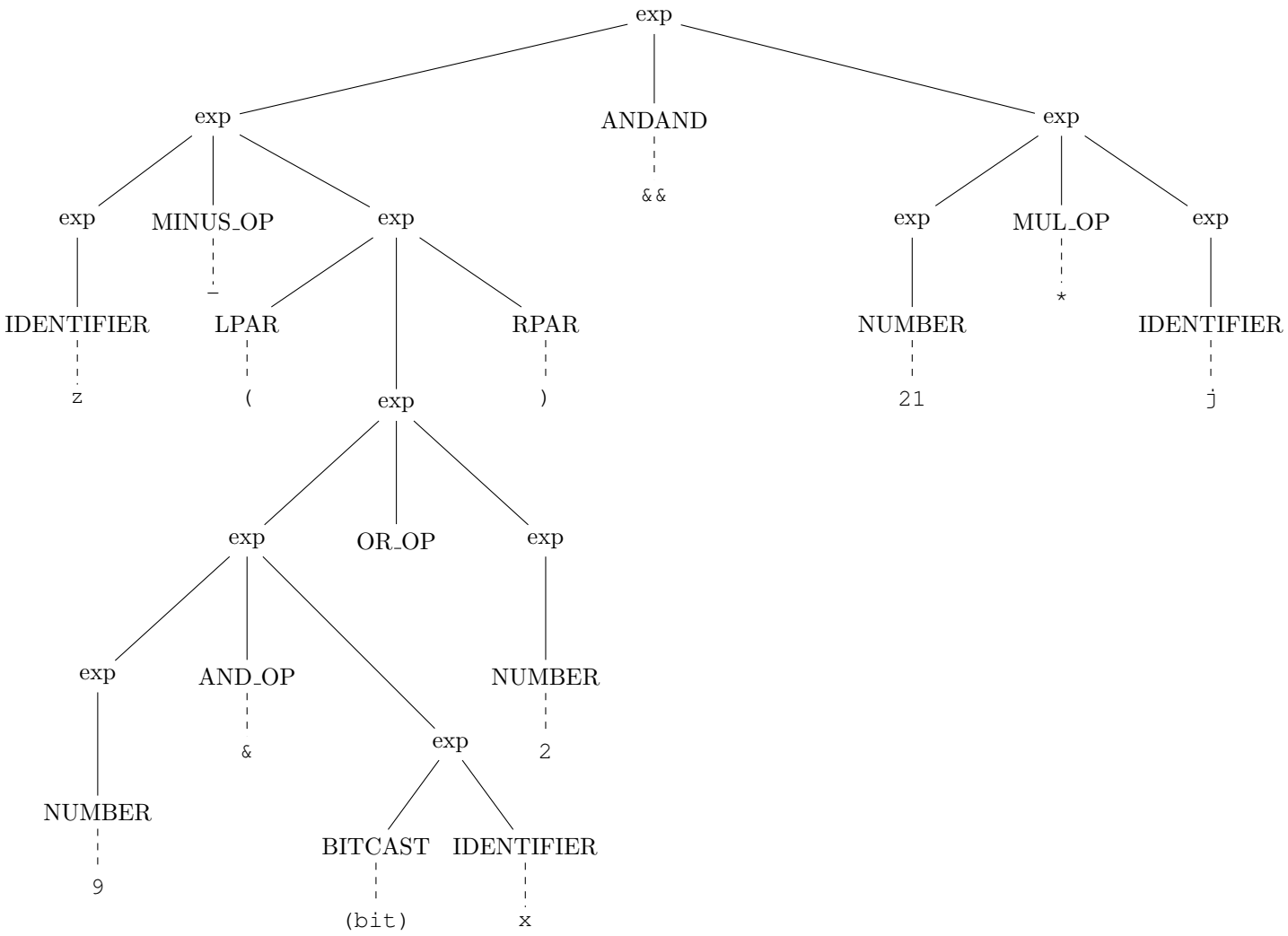
[1] Time 60'. Textbooks and notes can be used.
Pencil writing is allowed. Write your name on any additional sheet.

1. Define the tokens (and the related declarations in **Acse.lex** and **Acse.y**). (1 points)
2. Define the syntactic rules or the modifications required to the existing ones. (3 points)
3. Define the semantic actions needed to implement the required functionality. (21 points)
   The solution is in the attached patch.

4. Given the following `Lance` code snippet:

```
z - (9 & (bit)x | 2) && 21 * j
```

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



5. (**Bonus**) Describe how to modify your solution to extend **(bit)** operator to deal with generic expressions.