Formal Languages and Compilers
Proff. Breveglieri and Morzenti
Written exam[1]: laboratory question
14/06/2021

SURNAME:..
NAME:.. Student ID:................
Course: ○ Laurea Magistrale ○ 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 of the syntactic analyser (bison input), and any other source file required to extend the Lance language with the multisplice operator. The multisplice operator performs a bitwise slicing of a scalar value, described by a comma-separated list of disjoint ordered ranges. Bit indices are zero-based and inclusive, e.g., a range of `5-8` splices from bit 5 to bit 8. The range boundaries must be pairs of ordered constant integers, defining disjoint intervals, if the intervals are not ordered or not disjoint, a syntax error is returned. If the defined ranges exceed the representation power of the underlying microarchitecture, the behaviour is not defined.

The following excerpt of code exemplifies the multislice operator.

```
int x = 524280;
//                 0b00000000000001111111111111111000

write(x.m<0-2,19-25>m);
// Writes 0 =      0b00000000000000000000000000000000
write(x.m<3-10>m);
// Writes 2040 =   0b00000000000000000000011111111000
write(x.m<13-18>m);
// Writes 516096 = 0b00000000000001111110000000000000
write(x.m<3-10,13-18>m);
// Writes 518136 = 0b00000000000001111110011111111000
```

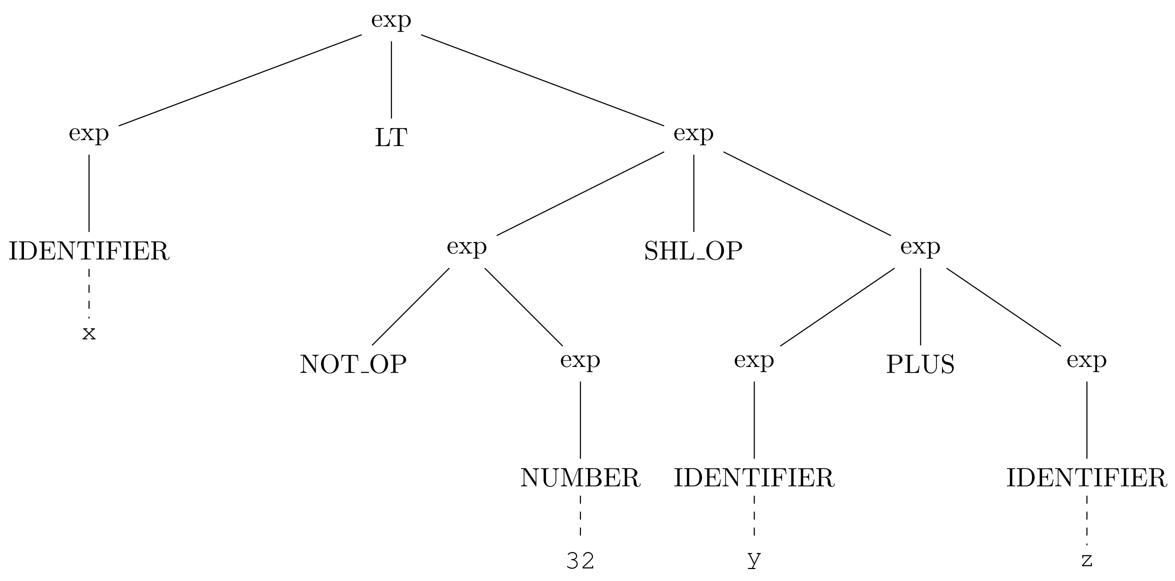
[1]Time 60'. The only material allowed during the exam is the ACSE reference header. Pencil writing is allowed. Write your name on any additional sheet.

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. (2 points)
3. Define the semantic actions needed to implement the required functionality. (22 points)
   The solution is in the attached patch.

4. Given the following `Lance` code snippet:

```
x < ! 32 << y + z
```

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



5. (**Bonus**) Explain how you would modify the implementation shown in question 3 in order to allow the use of non-constant range delimiters, as shown in the following statement:

```
int x = 524280, y = 4;
//                 0b00000000000001111111111111111000

write(x.m<2-y>m);
// Writes 24 =     0b00000000000000000000000000011000
y = y + 4;
write(x.m<2-y>m);
// Writes 504 =    0b00000000000000000000000111111000
```