2.1 Can you determine the reasons for the results obtained?

We have not defined a priority order for operations, so if there are no parentheses the expression is always solved from right to left.

2.2 What is the reason for that? What solutions can you think of? Analyse the code of the lexical analyser (yylex).

The reason for it is this little code snippet in yylex which allows an infinite amount of spaces without changing the result:

```
do {
c = getchar () ;
} while (c == ' ') ;
```

If we eliminate it, writing a space will result in a syntax error. A possible solution would be to include the blank space character in the grammar and allow it before and after the operator, before and after the expression and before and after the parenthesis, but never in between numbers.

We solved it in our new version of calc2.y.

2.3. There is a small bug as defined in the expression. Do you know in which cases it appears and why? Try different types of expressions.

The small bug relates to only being able to write as the first expression a number and not something inside parentheses. The bug appears whenever the first expression before the operator is in between a parenthesis, for example (1+2)/3. This can be easily fixed by moving the production rule related to parentheses to the operand non-terminal.

2.4 Can you say why it is valid in the first case and not in the second?

Because there is no ambiguity from the user at the level of the string of numbers. In the second example the computer doesn't know if the user will continue writing, but in the first example the user has already finished writing and the ambiguity can be resolved.

2.5 Describe what the yylex() function does

This function is dedicated to reading the text string character by character. Initially it ignored blank spaces when giving the next character, but after our solution to problem 2.2 it just returns the next character regardless of what it is.