PROCESSOR LANGUAGES: LAB 3

Question 1

The hierarchy of operations in the code still remains unclear to the bison. It reads the operations from right to left, so when the multiplication is on the left side, the result will be incorrect.

Question 2

We have changed ‘operator’ by ‘expression’ and added the production rules of the non-terminal symbol of operator in that of expression. With this modification, it now understands the order of the operations and no longer provides incorrect results. Below is the new modificated code

```
expresion:
              | expresion  '+'expresion   { $$  = $1 + $3; }
              | expresion '-' expresion  { $$  = $1 - $3; }
              | expresion  '*' expresion  { $$  = $1 * $3; }
              | expresion  '/' expresion  { $$  = $1 / $3; }
              | '+' NUMERO %prec SIGNO_UNARIO             { $$ = $2; }
              | '-' NUMERO %prec SIGNO_UNARIO             { $$ = -$2; }
              | NUMERO                                    { $$ = $1; }
              | '(' expresion ')'                         { $$ = $2; }
              ;
%%
```

Question 3 & 4

Firstly, we add %token VARIABLE. Then we have added this line in axiom:

VARIABLE '=' expresion '\n' { update($1, $3); printf("Asignado %s = %lf\n", $1, $3); }

which gives you permission to assign a value to a variable.

Finally, we have also included VARIABLE as a production rule in expression

```
| NUMERO                                    { $$ = $1; }
| VARIABLE                                  { $$ = $1}
```

Question 5

We have added this so that we can match regular expressions with variable names.

```
[a-zA-Z][a-zA-Z0-9]*     { yylval = strdup(yytext); return VARIABLE; }
"="                      { return '='; }
```

Question 6

When implementing a parser with the latest modifications, we observe that the program can not work with them, as more functions are needed so the implemented functions do not create ambiguity.