Project

E0025323

The syntax used is as suggested in the project document, with the following modifications:

- No semicolons between the tags in a relationship type. I think that a space is sufficient to differentiate between them, since the tags will never be complex.
- Semicolon after every element in the relation (including the last one).
- Operator for join is "@". I prefer single sign operators.
- Operator for projection is "#". Same argument as above.
- The tags in a projection is only separated by a space, they're not provided as a tuple like structure. I think this is much cleaner.

My source is based on that of lab 7. The parsing bit is modified in sPL_parser (for rules), and sPL_lexer and sPL_token (for lexing). The type checking is done in sPL_type, and the running is done in sPL_compile. Everything is compiled using the command 'make proj'.
The type checking and parsing is fully complete. To only use test these features, make sure line 317 in sPL_compile is 'if false then'. All the samples are in the samples folder.

The test cases that try these out are:

- sp_ex1.spl (correct)
- sp_ex2.spl (incorrect, second row contain one integer too much).
- sp_ex3.spl (incorrect, the first element of the second row is of the wrong type)
- sp_ex4.spl (correct)
- sp_ex9.spl (correct, with let construct and join operation)
- sp_ex10.spl (correct, with let construct and proj operation).

To run this, I implemented a simple interpreter for the two operations. Make sure like 317 is 'if true then'.

The test cases to try this out are:

- sp_ex5.spl (projection, no duplicates)
- sp_ex6.spl (projection, duplicates)
- sp_ex7.spl (join, duplicates)

- sp_ex8.spl (join, no duplicates)

I didn't have time to fully develop the compiler to support running relations with everything

else. However, this should provide enough support.