COL226: Programming Languages

Assignment: A `BigInt` Calculator Package

Refer to the ML-Lex and ML-Yacc manuals available on the web. Section 11.1 describes the grammar for a simple calculator language that can be scanned and parsed for integer computations. However this grammar is for an LR parser and may not be amenable to recursive descent parsing.

Also refer to your `bigint` package. Now modify the calculator grammar so as to make it LL(1) parseable and allows the use of all the functions of the `bigint`. Specifically,

1. Design an LL(1) grammar for a calculator language which generates code for the `bigint` package.
2. Incorporate the usual rules of precedence and associativity which govern arithmetic expressions.

   Warnings.

   (a) Do not make any prefix operator infix if it is not usually used in infix form. Leave all such operators in prefix form.

   (b) Do not create your own associativity or precedence rules. Use the usual preceence and associativity that is normally associated with each infix operator.

   (c) All the functions/operators define in the signature of the `bigint` package must be available to the user of your calculator package (including functions like *len*, *lenLt*, . . . which are not usually available in a calculator language.

   (d) Since some of these functions may return values of type `int` rather than `bigint` it might be a good idea to distinguish between the `bigint` operations and `int` operations by using the infix symbols given in the signature of `functor BigInt`.

   (e) Do not change the names of any of the `bigint` operators/functions in your package.
3. Write a **recursive descent parser** for the calculator language
4. Write an interpreter for the code so generated so that your entire package can be used like the `bc` program on linux.
5. Write a README file which describes the LL(1) grammar and also the means of operating your package.
6. `tar gzip` all the files including the README file into a file called `<your-rollno>.tgz` and upload it on moodle.