Module 1 Peer Review – Tomás Belmar

Module 1 consisted of two different parts for this assignment. One part was to build a lexer and evaluator for a calculator, and the second part was to build a lexer and parser for Cigrid, a language that can be described as a subset of C. For both parts of these assignments, this student chose to use Ocaml, the course's recommended programming language. For the calculator, the student starts by making a homemade lexer. They hav three different regular expressions: one to detect whether an expression is valid, one to tokenize the expression, and one to detect (or rather, ignore) whitespace. They create two helper functions, standardize_whitespace which removes whitespace given an input, and is_valid_input which detects whether the input given is a valid expression. Then they use a function tokenize_input that splits the expression into different tokens. His parser uses different functions to parse expressions, terms, and factors. They decide to include the "prime" versions of these functions inside them rather than separate from them, which makes the code cleaner and easier to read. Finally, they combine all of this in his main file so that standard input can be assessed and evaluated as a mathematical expression. For their Cigrid lexer and parser this student goes a different route. Instead of manually lexing the tokens, they use a .mll file (Ocamllex) to do most of the work for him. Therefore, all they have to do is define what characters are assigned to what types (that they create in another file), and Ocamllex will do the hard parts. Something they did differently from me was they used regular expressions to match identifiers, multi line comments, etc. Their parser also leverages Menhir so they don't have to do most of the hard work. Instead, they can just define their tokens and their grammar, and Menhir does all the hard parts. This student seems to have gone for the higher grades, and has very organized and easy to read code. It seems that their multiline comment code may not capture every single multiline comment, as it requires that multiline comments don’t have asterisks inside them. This student seems to have tested their solutions very well as there are several test files laying in their directory.

Questions

1. Was it necessary to separate your “If” and “If Else” statements into different types of AST Nodes?
2. What is the point of the “Nil” types?
3. Where is the “Arg” package coming from in main.ml? You’re not importing it from anywhere?