A3: Interpretation

Purpose of Assignment:

- Implement a complete interpreter for an extended version of the calculator language.
- Catch the following dynamic errors:
    - Unexpected end of input
    - Non-numeric input
    - Use of an uninitialized variable
    - Divide by zero

Approach:

The method that made the most sense to me was to generate the AST by matching every element of the parse tree and then essentially generating the AST from the leaves of that parse tree. This process was simply repeated for most of the “ast_ize” functions.

The next part of the assignment was to interpret the AST. I simply evaluated the AST from left to right. What I did was simply separate the head of the statement list from the tail and interpret that statement. From there I would interpret the smaller elements of that statement such as an expression. Evaluating expressions involved recursively calling “interpret_expr” on the left hand side and right hand side of the expression, until it was broken up into it’s most basic elements (id, num). from there it would return the value back up the call stack. The other interpret functions were implemented similarly to “interpret_s”

How to run: (“~” indicates command line prompt)

```
~ ocamlc str.cma interpreter.ml
~ ./a.out
```

Jean-Marc Boullianne
jboullia@u.rochester.edu

Test Output:

```
[jboullia@cycle1 A3]$ ./a.out
10 5

2 3 5 7 11 13 17 19 23 29

3 divide by zero

foo: symbol not found

3 divide by zero

unexpected end of input

3 divide by zero
```

Included Files:

- Interpreter.ml