CS320 Programming Languages
Exercise #3: MVAE

The goal of Exercise #3 is to implement the interpreter of `MVAE`, which is the extended version of `VAE` that deals with multiple values.

1 Supporting Multi-Value

The first thing we should fix is that we still define a `Num` with a single numeric value and use it as the result of `run`.

One way to resolve this would be to add a new `case class` called `Nums` to our AST definition. But this would require reworking new cases for it in a few places. So instead, we will choose an easier solution: just change the existing `Num` so instead of holding a single integer it will hold a `List[Int]`.

We need to fix the arithmetic operators. This is a bit tricky, since each of them receives two inputs that are both lists, and they should apply the operator on each pair from these two inputs, and collect a list of all of the results. So to make it easy, here is a skeleton of a utility function that will do this work. It is near-complete, and you have a tiny hole to fill:

```
// applies a binary numeric function on all combinations of integers from
// the two input lists, and return the list of all of the results
def binOp(
  op: (Int, Int) => Int,
  ls: List[Int],
  rs: List[Int]
): List[Int] = ls match {
  case Nil => Nil
  case l :: rest =>
    def f(r: Int): Int = ??? // TODO: complete this function
    rs.map(f) ++ binOp(op, rest, rs)
}
```

Don't forget to add tests that demonstrate that this works: that using with to bind a name to a multi-valued expression works as expected. Here are some tests that should work once you're done with this part:

```
test(run("(3 + 7)"), List(10))
test(run("(10 - (3, 5))"), List(7, 5))
test(run("{ val x = (5 + 5); (x + x) }"), List(20))
```

2 Adding More Arithmetic Operators

Next, add two arithmetic operators, `min` and `max` that take three expressions and return the minimum and the maximum list of integers, respectively, to `MVAE`:

```
test(run("min(3, 4, 5)"), List(3))
test(run("max((1 + 2), 4, 5)"), List(5))
test(run("min((1, 4), (2, 9), 3)"), List(1, 1, 2, 3))
test(run("max((1, 6), (2, 5), (3, 4))"), List(3, 4, 5, 5, 6, 6, 6, 6))
```

3 Formal Definitions

3.1 Concrete Syntax

The concrete syntax of `MVAE` is written in the extended Backus-Naur form (EBNF). Note that `{ }` denotes a repetition of zero or more times.

```
expr  ::=  num
         |  "( ")"
         |  "(" num { "," num } ")"
         |  "(" expr "+" expr ")"
         |  "(" expr "-" expr ")"
         |  "{" "val" id "=" expr ";" expr "}"
         |  id
         |  "min" "(" expr "," expr "," expr ")"
         |  "max" "(" expr "," expr "," expr ")"
```

3.2 Abstract Syntax

For abstract syntax, we are using $\cdots$ to denote repetitions.

$$
\begin{array}{rcl}
e & ::= & (n, \cdots, n) \\
 & | & e + e \\
 & | & e - e \\
 & | & \mathsf{val}\ x = e;\ e \\
 & | & x \\
 & | & \mathsf{min}(e, e, e) \\
 & | & \mathsf{max}(e, e, e) \\
n, m, l & \in & \mathbb{Z} \\
x & \in & \mathit{Var} \\
\sigma & \in & \mathit{Var} \to \{(n_1, \cdots, n_i) \mid i \in \mathbb{N} \wedge n_1, \cdots, n_i \in \mathbb{Z}\}
\end{array}
$$

3.3 Operational Semantics

$\boxed{\sigma \vdash e \Rightarrow (n, \cdots, n)}$

$$\sigma \vdash (n_1, \cdots, n_i) \Rightarrow (n_1, \cdots, n_i)$$

$$\frac{\sigma \vdash e_1 \Rightarrow (n_1, \cdots, n_i) \quad \sigma \vdash e_2 \Rightarrow (m_1, \cdots, m_j)}{\sigma \vdash e_1 + e_2 \Rightarrow (n_1 + m_1, n_1 + m_2, \cdots, n_i + m_{j-1}, n_i + m_j)}$$

$$\frac{\sigma \vdash e_1 \Rightarrow (n_1, \cdots, n_i) \quad \sigma \vdash e_2 \Rightarrow (m_1, \cdots, m_j)}{\sigma \vdash e_1 - e_2 \Rightarrow (n_1 - m_1, n_1 - m_2, \cdots, n_i - m_{j-1}, n_i - m_j)}$$

$$\frac{\sigma \vdash e_1 \Rightarrow (n_1, \cdots, n_i) \quad \sigma[x \mapsto (n_1, \cdots, n_i)] \vdash e_2 \Rightarrow (m_1, \cdots, m_j)}{\sigma \vdash \mathsf{val}\ x = e_1;\ e_2 \Rightarrow (m_1, \cdots, m_j)}$$

$$\frac{x \in \mathit{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

$$\frac{\sigma \vdash e_1 \Rightarrow (n_1, \cdots, n_i) \quad \sigma \vdash e_2 \Rightarrow (m_1, \cdots, m_j) \quad \sigma \vdash e_3 \Rightarrow (l_1, \cdots, l_k)}{\sigma \vdash \mathsf{min}(e_1, e_2, e_3) \Rightarrow (\mathit{min}(n_1, m_1, l_1), \mathit{min}(n_1, m_1, l_2), \cdots, \mathit{min}(n_i, m_j, l_k))}$$

$$\frac{\sigma \vdash e_1 \Rightarrow (n_1, \cdots, n_i) \quad \sigma \vdash e_2 \Rightarrow (m_1, \cdots, m_j) \quad \sigma \vdash e_3 \Rightarrow (l_1, \cdots, l_k)}{\sigma \vdash \mathsf{max}(e_1, e_2, e_3) \Rightarrow (\mathit{max}(n_1, m_1, l_1), \mathit{max}(n_1, m_1, l_2), \cdots, \mathit{max}(n_i, m_j, l_k))}$$