CS 496 – Quiz 3 – 11/Mar/21

Exercise 1

Consider the language LET+MAX, an extension of the LET-language with an operation for obtaining the maximum of two numbers. Its concrete syntax is given below:

⟨Expression⟩ ::= ⟨Number⟩
⟨Expression⟩ ::= ⟨Identifier⟩
⟨Expression⟩ ::= ⟨Expression⟩⟨BOp⟩⟨Expression⟩
⟨Expression⟩ ::= `zero?`(⟨Expression⟩)
⟨Expression⟩ ::= `if` ⟨Expression⟩ `then` ⟨Expression⟩ `else` ⟨Expression⟩
⟨Expression⟩ ::= `let` ⟨Identifier⟩ `=` ⟨Expression⟩ `in` ⟨Expression⟩
⟨Expression⟩ ::= `max`(⟨Expression⟩, ⟨Expression⟩)
⟨Expression⟩ ::= (⟨Expression⟩)

⟨BOp⟩ ::= `+` | `-` | `*` | `/`

Only one production in the grammar is new, the one for `max`. Examples of programs in LET+MAX are:

1. `max(2,3)`. Should evaluate to `Ok (NumVal 3)`
2. `max(max(7,9),3)`. Should evaluate to `Ok (NumVal 9)`.
3. `max(zero?(4),11)`. Should evaluate to `Error "Expected a number!"`.
4. The program `let x = 34 in max(x,5)` should evaluate to `Ok (NumVal 34)`.

You are asked to extend the interpreter for LET to LET+MAX, so that `eval_expr` is capable of executing expressions involving `max`.

```
let rec eval_expr =
  fun e en ->
  match e with
  | Int n -> return (NumVal n)
  | Var id -> apply_env en id
  | ...
  | Max(e1,e2) ->
      failwith "implement me"
```