CS 511 – Quiz 6: Sequential Erlang

30 March 2023

Names:
Pledge:

Exercise 1

Implement a function `eval/2` that given a *calculator expression* and an *environment* (a map from identifiers to integers; see example below), returns the result of evaluating the expression. Calculator expressions are modeled using tuples, atoms and numbers. For example, the expression "3+(x/4)" is represented by the calculator expression:

```
{add,
 {const,3},
 {divi,
  {var,"x"},
  {const,4}}}.
```

Other examples are given in the stub. The grammar for calculator expressions is as follows:

```
<exp> ::= {var,<string>}
        | {const,<integer>}
        | {add,<exp>,<exp>}
        | {sub,<exp>,<exp>}
        | {mul,<exp>,<exp>}
        | {divi,<exp>,<exp>}
```

You must complete the given stub (file `calc.erl`). This stub should include an `eval/2` function. This function should either return a *value* or throw an *exception*:

- A value is an expression of the form indicated below. Note that values are not integers, they are tuples!

  ```
  <val> ::= {val,<integer>}
  ```

- There are two possible exceptions:
  - `unbound_identifier_error`: the variable is not in the environment.
  - `division_by_zero_error`: there is a division by zero.

  You throw an exception by placing these atoms as an argument to the `throw/1` function. As follows:

  - `throw(unbound_identifier_error)`
  - `throw(division_by_zero_error)`

  In this quiz you will only throw exceptions, not handle them.

Consider the following environment and sample calculator expressions (they are included in the stub):

```
env() -> #{"x"=>3, "y"=>7}.

e1() ->
    {add,
     {const,3},
     {divi,
      {var,"x"},
      {const,4}}}.

e2() ->
    {add,
     {const,3},
     {divi,
```

```
      {var,"x"},
      {const,0}}}.

e3() ->
    {add,
     {const,3},
     {divi,
      {var,"r"},
      {const,4}}}.
```

Then, in the shell we will have:

```
11> calc:eval(calc:e1(),calc:env()).
{val,3}
12> calc:eval(calc:e2(),calc:env()).
** exception throw: division_by_zero_error
     in function  calc:eval/2 (calc.erl, line 118)
     in call from calc:eval/2 (calc.erl, line 104)
13> calc:eval(calc:e3(),calc:env()).
** exception throw: unbound_identifier_error
     in function  calc:eval/2 (calc.erl, line 99)
     in call from calc:eval/2 (calc.erl, line 115)
     in call from calc:eval/2 (calc.erl, line 104)
14>
```

You will need to lookup the `maps:find/2` operation on maps. Also, Erlang **`case`** expressions will be useful to analyze the result of the call to`maps:find/2`.

Also useful is the matching construct. For example, `{val,N}= eval(E1,Env)` will bind the number component of the tuple obtained from evaluating `eval(E1,Env)` to `N`.

Submit your file `calc.erl` via Canvas.