Homework Assignment 5

Any automatically graded answer may be manually graded by the instructor. Submissions are expected to only use functions taught in the course. If a submission uses a disallowed function, that exercise can get zero points. Excluding promises, *all functions that mutate values are disallowed* (mutable functions usually have a `!` in their name).

Language λ_D (now with `define`)

Note: This section requires functions and structures defined in file ***`hw5-util.rkt`****. Tests can be found in file* ***`test-hw5.rkt`****.*

1. *Your goal is to implement the evaluation of expressions using a mutable environment, notation* $e \Downarrow_E v$. Implement function `(d:eval-exp mem env exp)`, whose contract is defined as

   ```
   (-> mem? handle? d:expression? eff?)
   ```

   where memory `mem` is a heap of frames, the environment `env` is a handle to a frame in the memory, and the expression `exp` is being evaluated. Function `d:eval-exp` must return an effect whose state is the possibly modified heap and the result is the evaluated value.

2. *Your goal is to implement the evaluation of terms using a mutable environment, notation* $t \Downarrow_E v$. Implement function `(d:eval-term mem env term)`, whose contract is defined as

   ```
   (-> mem? handle? d:term? eff?)
   ```

   where memory `mem` is a heap of frames, the environment `env` is a handle to a frame in the memory, and the term `term` is being evaluated. Function `d:eval-term` must return an effect whose state is the possibly modified heap and the result is the evaluated value.

Manually graded questions

3. **Manually graded.** Considering how definitions work in Racket and our specification of Language λ_D, *distinguish the variable binding semantics of Language* λ_D *from Racket. Give an example of a program that behaves differently in* λ_D *and in Racket.* We are not interested in features that are implemented in one language but are not in another (say the lack of promises), so only consider programs whose syntax is the same for both λ_D and Racket.