CS 496: Extra Homework

1 Assignment Policies

Collaboration Policy. Homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

2 Assignment

This assignment consists in extending REC to allow for mutually recursive function definitions. The resulting language will be called REC-M. It modifies the concrete syntax for `letrec` as follows. The production

$$\langle\textsf{Expression}\rangle \quad ::= \quad \texttt{letrec}\ \langle\textsf{Identifier}\rangle = \langle\textsf{Expression}\rangle\ \texttt{in}\ \langle\textsf{Expression}\rangle$$

in REC is replaced with:

$$\langle\textsf{Expression}\rangle \quad ::= \quad \texttt{letrec}\ \{\ \langle\textsf{Identifier}\rangle = \langle\textsf{Expression}\rangle\}^*\ \texttt{in}\ \langle\textsf{Expression}\rangle$$

in REC-M. The expression $\{\ \langle\textsf{Identifier}\rangle = \langle\textsf{Expression}\rangle\}^*$ above means that there may be 0 or more declarations. Here is an example of a valid program in REC-M:

```
letrec
  even(x) = if zero?(x)
              then 1
              else (odd -(x,1))
  odd(x)  = if zero?(x)
              then 0
              else (even -(x,1))
in (odd 99)
```

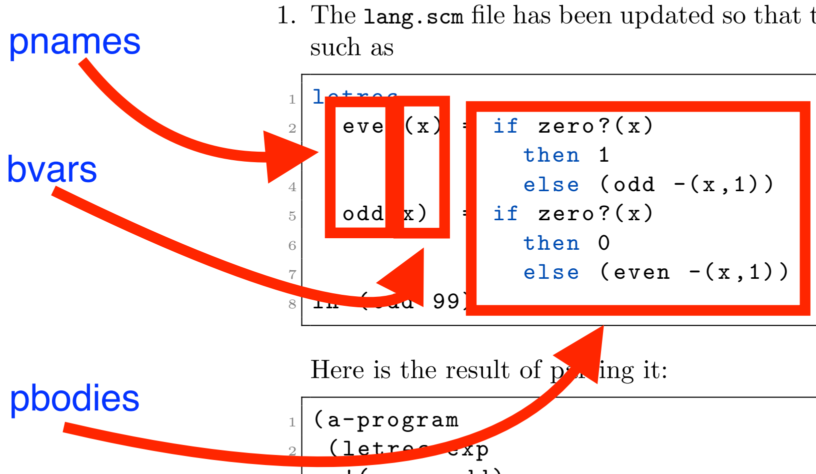
Evaluating that expression should produce the result `(num-val 1)`, meaning that 99 is indeed odd. If we replace 99 in the code above with 98 and evaluate the resulting expression, this time we should get `(num-val 0)` as a result. This is correct since 98 is not an odd number.

Note that the above expression is not syntactically valid in REC. To see this, try running it in the interpreter for REC.

3 Implementing REC-M

To facilitate the process of implementing REC-M a stub has been provided for you in Canvas. This stub has been obtained by taking the interpreter for REC and applying some changes. Here is a summary of the changes:

1. The `lang.scm` file has been updated so that the parser is capable of parsing expressions such as



```
letrec
  even(x) = if zero?(x)
              then 1
              else (odd -(x,1))
  odd(x) = if zero?(x)
              then 0
              else (even -(x,1))
in (odd 99)
```

   Here is the result of parsing it:

```
(a-program
 (letrec-exp
  '(even odd)
  '(x x)
  (list
   (if-exp (zero?-exp (var-exp 'x)) (const-exp 1) (call-exp
    (var-exp 'odd) (diff-exp (var-exp 'x) (const-exp 1))))
   (if-exp (zero?-exp (var-exp 'x)) (const-exp 0) (call-exp
    (var-exp 'even) (diff-exp (var-exp 'x) (const-exp 1)))))
  (call-exp (var-exp 'odd) (const-exp 99))))
```

   Note that the first three arguments to `letrec-exp` are now lists.

2. The `environment` datatype has been updated by commenting out the variant for `extend-env-rec` and replacing it with one for `extend-env-rec*`. Note the use of `list-of` here.

```
(define-datatype environment environment?
 (empty-env)
 (extend-env
  (bvar symbol?)
  (bval expval?)
  (saved-env environment?))
 ;    (extend-env-rec  ;; superseded by extend-env-rec*
;       (id symbol?)
;       (bvar symbol?)
;       (body expression?)
;       (saved-env environment?))
(extend-env-rec*
 (proc-names (list-of symbol?))
 (b-vars (list-of symbol?))
 (proc-bodies (list-of expression?))
 (saved-env environment?)))
```

You will have to update

1. `apply-env` in the file `environments.scm`. It currently reads as follows:

```
(define apply-env
  (lambda (env search-sym)
    (cases environment env
      (empty-env ()
        (eopl:error 'apply-env "No binding for ~s" search-sym))
      (extend-env (var val saved-env)
        (if (eqv? search-sym var)
          val
          (apply-env saved-env search-sym)))
      (extend-env-rec (p-name b-var p-body saved-env)
        (if (eqv? search-sym p-name)
          (proc-val (procedure b-var p-body env))
          (apply-env saved-env search-sym))))))
```

   It should deal with `extend-env-rec*` rather than `extend-env-rec`. Also, you can introduce auxiliary functions, if required.

2. `value-of` in the file `interp.scm`. The case that must be updated is this one:

```
(letrec-exp (p-name b-var p-body letrec-body)
       (value-of letrec-body
                 (extend-env-rec p-name b-var p-body env)))
```

4 Submission instructions

This assignment is not to be submitted.