```
;; Top-Level for the Scheme Interpreter.

;; This file must be loaded before scheme.scm.
;; For loading the file, you can either use the command
;;      ,load top-level.scm
;; or
;;      (load "top-level.scm")
;; or
;;      (load 'top-level)
;; Similarly for loading printenv.scm and scheme.scm.

;; You can get in and out of scheme48 and your interpreter as follows:
;;
;; % scheme48
;; > ,load top-level.scm
;; > ,load printenv.scm
;; > ,load scheme.scm
;; > (scheme)
;; Scheme 4101> (+ 1 2
;; Scheme 4101> ^
;; > ,exit

;; Rename the built-in versions of apply and eval.
(define builtin-apply apply)
(define builtin-eval eval)


;; Define the empty global environment.
;; '(()) doesn't work with set-car!, it must be (list '())
;;
;; With global-environment being a global variable, it is difficult to
;; run the Scheme interpreter on top of itself.  An alternative would
;; be to use let to bind builtin-apply, builtin-eval, and global-environment.
(define global-environment (list '()))


;; Clear the environment.
(define (clear-env)
  (set! global-environment (list '())))


;; The top-level read-eval-print loop.
;; Use CNTL-D to exit the interpreter.
(define (scheme)
  (display "Scheme 4101> ")
  ; Parse the input.
  (let ((input (read)))
    (if (eof-object? input)
        (newline)
        (begin
          ; Evaluate the expression and print the result.
          (write (eval input global-environment))
          (newline)
          (scheme)))))


;; Load a file into the global environment.
;; Use (loadenv "foo.scm") to load expressions from file foo.scm;
;; loadenv cannot be used before scheme.scm is loaded.
(define (loadenv file)
  (define (loadenv1 file port env)
    (let ((input (read port)))
      (if (eof-object? input)
          (begin
            (close-input-port port)
            (display file)
            (newline))
          (begin
            (eval input env)
            (loadenv1 file port env)))))

  (loadenv1 file (open-input-file file) global-environment))
```