```
;;;----------------------------------------------------------------------------
;;; CLOSURIZE.SCM
;;;
;;; A closure converter that makes flat closures for all LAMBDAs and FUNRECs.
;;; Writing CLOSURIZE-FUNREC is left as an exercise.
;;;
;;;----------------------------------------------------------------------------

(define (closurize node)
(cond ((application-node? node) (closurize-application node))
((named-primop-node? 'procedure? node) (closurize-procedure? node))
((lambda-node? node) (closurize-lambda node))
((funrec-node? node) (closurize-funrec node))
(else (subnode-map closurize node))))

(define (closurize-application node)
`(CALL-CLOSURE ,(closurize (call-rator node))
        ,@(map closurize (call-rands node))))

(define (closurize-procedure? node)
`(PRIMOP CLOSURE? ,@(map closurize (primop-args node))))

(define (closurize-lambda node)
(let ((formals (lambda-formals node))
(body (lambda-body node))
(frees (free-vars node))
(closure-var (make-var (fresh-name 'closure))))
`(PRIMOP CLOSURE
    (LAMBDA (,closure-var ,@formals)
        ,(rewrite (list->set frees)
            ;; Ref-rewriting procedure
            (lambda (var)
            (make-primop 'closure-ref
                (list closure-var
                ;; Need 1+ to pass over code
                (1+ (position var frees)))))
            ;; SET!-rewriting procedure
            (lambda (var body)
            (make-primop 'closure-set!
                (list closure-var
                ;; Need 1+ to pass over code
                (1+ (position var frees))
                body)))
            (closurize body)))
    ,@frees)))
```